Preconditioned Newton methods to approximate solutions of the Reynolds averaged Navier-Stokes equations

Habilitationsschrift

am Fachbereich Mathematik und Naturwissenschaften

der Universität Kassel

vorgelegt von

Dr. Stefan Langer

Gutachter:	Prof. Dr. Andreas Meister
Zweitgutachter:	Prof. Dr. Thomas Sonar
Drittgutacher:	Prof. Dr. Nicolas R. Gauger

Abstract

We consider nonlinear agglomeration multigrid methods tailored to the efficient and robust solution of the Reynolds averaged Navier-Stokes equations for the simulation of high Reynolds number turbulent flows.
In the first part of this thesis we introduce the Reynolds averaged Navier-Stokes equations together with established turbulence models required to determine the eddy viscosity arising from Bousinessq' assumption to close the set of equations. A complete nondimensionalization of the equations is presented. Finally, corresponding boundary value problems are presented. It is the goal of the thesis to find approximate solution to these problems.
In a second part of this thesis the discretization strategy of the equations is presented in detail, in particular concerning boundary conditions. Moreover, a detailed description of the computation of derivative of the discretized equations is presented. Emphasis is appointed to this topic since the construction and realization of exact and approximate derivatives are of major importance to formulate and derive efficient and robust solution algorithms.
These solution algorithms are based on nonlinear agglomeration multigrid methods including implicit Runge-Kutta smoothers. The general derivation of these methods presented in this thesis allows to interpret the implicit Runge-Kutta smoothers as stabilized Newton methods. On the other hand, the application of certain simplifications allows to generate almost all solution methods well known in the world of computational fluid dynamics. This hierarchy of methods gives insight in the potential and shortcomings of several methods established and in daily use in many computer codes approximating solutions of the Reynolds averaged Navier-Stokes equations. To get a deeper understanding of the methods an analysis tool is suggested to evaluate certain smoothers.
Finally, several examples are presented to show the potential and shortcomings of certain methods derived in this thesis. Comparisons are done with respect to both robustness and efficiency of the scheme. In dealing with large scale problems the challenge of scalability on high performance clusters is discussed.
A further important point in this thesis is the accuracy of the suggested discretization scheme. Since for the Reynolds averaged Navier-Stokes equations no analytic solutions are available, classical basic test cases as well as those of industrial relevance are investigated with respect to mesh refinement. Methods are applied to get insight into the accuracy obtained on a sequence of meshes. With respect to this topic, different approaches towards the incompressible limit are presented and compared with respect to reliability and accuracy.

Danksagung

An dieser Stelle möchte ich die Gelegenheit nutzen, mich bei allen zu bedanken, die es mir ermöglicht haben, diese Habilitationsschrift zu verfassen.

Zuvorderst bedanke ich mich bei meinem Abteilungsleiter der Abteilung $C^2A^2S^2E$, Herrn Prof. Dr. Norbert Kroll, und beim Institutsleiter des Instituts für Aerodynamik und Strömungstechnik des Deutschen Zentrums für Luft-und Raumfahrt, Herrn Prof. Dr. Cord-Christian Rossow. Beide haben mir während meiner Tätigkeit immer wieder Freiräume eingeräumt, mich mit neuen Herausforderungen auseinanderzusetzen und Ideen zu entwickeln, um diese zu lösen. Besonders möchte ich die freundschaftliche, immer von neuem motivierende Art und Weise, mit der Herr Prof. Dr. Norbert Kroll mich immer wieder ermutigte, neue Aufgaben und Probleme anzugehen, hervorheben. Ich danke ihm sehr dafür.

Auch der Dank gegenüber Herrn Dr. Charles Swanson sei an dieser Stelle ausdrücklich erwähnt. Seine beeindruckenden Kenntnisse und Hintergründe im Fachgebiet der numerischen Strömungsmechanik und Turbulenzmodellierung waren eine stete Quelle der Inspiration.

Des Weiteren danke ich meinem sehr guten Freund und Kollegen Tobias Leicht. Seine vielen Anregungen, Ideen und kritischen Nachfragen waren immer sehr inspirierend und Motivation, viele Dinge zu hinterfragen und weiter zu verbessern.

Herrn Prof. Dr. Andreas Meister vom Fachbereich Mathematik und Naturwissenschaften der Universität Kassel danke ich für die Betreuung und Begutachtung der Arbeit. Für mich war es sehr ermunternd und motivierend, dass er sofort nach Vorstellung der möglichen Inhalte bereit war, das Habilitationsverfahren von universitärer Seite zu begleiten. Insbesondere für die Möglichkeit der Wahrnehmung einer Vertretungsprofessur und die damit verbundenen Lehraufgaben, die mir viel Freude bereiteten, möchte ich mich sehr herzlich bedanken.

Bei Herrn Prof. Dr. Thomas Sonar von der Universität Braunschweig und bei Herrn Prof. Dr. Nicolas R. Gauger von der Universität Kaiserslautern bedanke ich mich sehr herzlich für die weiteren Begutachtungen der Arbeit.

Mein größter Dank gebührt meiner Familie, meiner Frau Antje und meinen Kindern Lukas und Mareike. Ein nicht unerheblicher Anteil der vorliegenden Arbeit entstand am Schreibtisch zu Hause. Für die eingeräumte Zeit und das Verständnis für das zu Hause entstandene Chaos danke ich Antje und meinen Kindern sehr. Meine Familie, die ich über alles liebe, gibt den Inhalten einen Sinn. Sie sind es, die mich daran erinnern, immer wieder das Wichtige vom Unwichtigen abzugrenzen. Und häufig genug war es deutlich wichtiger und sinnvoller, die Arbeit zur Seite zu legen, um mit den Kindern zu spielen oder mit meiner Frau Antje etwas zu unternehmen, als weiter an dieser Arbeit zu schreiben.

Contents

Chapter 1

Introduction

The design of ecologically sensitive, low noise, and safe aircraft is the major challenge of the aircraft industry for the following decades. A key technology to reach these goals is the numerical simulation. To accurately predict aerodynamic properties of an aircraft the Euler and Navier-Stokes equations are in focus. These equations model the motion of a fluid in $\mathbb{R}^n$; and therefore, solutions to these equations are of major importance in giving answers to many classical physical questions. In our context, solutions of these equations give clarity and a deep insight into a better understanding to the interaction of a fluid surrounding an aircraft. Such a deep understanding is a key requirement for future aircraft to be significantly more efficient than existing ones. To be prepared for future challenges in the aircraft industry and other technical disciplines based on the understanding of fluid flows, it is the goal of this thesis to identify the main ingredients required to design reliable algorithms to approximate solutions of the Euler and Navier-Stokes equations.

To be more precise, within this thesis the focus is to approximate solutions of the so-called **R**eynolds **a**veraged **N**avier-**S**tokes (RANS) equations. These equations may be derived from the Navier-Stokes equations using time averaging of the variables. Solutions of these equations therefore represent time averaged states. Such time averaging leads to additional terms with additional unknowns. Mathematically, the resulting system of equations is more complex in the sense that for the additional unknowns additional equations are required. A significant simplification is attained by considering Boussinesq' eddy viscosity assumption. As a consequence of this assumption, the RANS equations are mathematically equivalent to the Navier-Stokes equations. With regards to content, the RANS equations differ in our context from the Navier-Stokes equations by a summand weighting the viscous contributions. This summand represents a nonnegative scalar function, which is called in general an eddy viscosity and is not explicitly given. To find closure conditions for this function either algebraic relations are established, or additional differential or integral equations are considered. Then, from the solution of these equations the eddy viscosity is determined. It is the topic of turbulence modeling to find suitable

closure conditions. Since the Navier-Stokes and the RANS equations are equivalent in form, within this thesis we often use these nomenclatures synonymously. But we attach great importance to the fact that either this summand is active or not.

Though progress has been made, even after several decades of developments and investigations for high Reynolds number viscous flows a severe loss of reliability, robustness and efficiency of solution methods for the compressible RANS equations can often be observed. Here, the wording solution method comprises both

a) the discretization strategy,

b) and the corresponding solution method.

The challenges to find solutions of this set of equations are manifold. And, the problems arising are not only observed for large-scale, complex industrial applications, but they also occur for many basic test cases well known from the literature. For example, the difficulty to find solutions of basic flow problems can be significantly increased when one increases systematically the number of degrees of freedom. Then, almost all solution methods known lose their efficiency. As long as there are severe issues to approximate solutions of basic test cases, it can be doubted that the solution methods considered so far can be used in industrial processes. But what are the reasons for these observations?

It is the topic of this thesis to develop, analyze and investigate solution methods designed for efficiently solving large scale nonlinear problems, which arise from the discretization of the Reynolds averaged Navier-Stokes equations.

Before we discuss several practical problems, it is worthwhile to note that up to now the uniqueness, existence and smoothness of a solution to the Navier-Stokes and even the Euler equations in $\mathbb{R}^3$ is an open problem. Due to the significance of the problem, the theoretical answer to existence and smoothness of solutions to the Navier-Stokes equations has been listed as one of the seven Millennium Prize Problems in mathematics that were stated by the Clay Mathematics Institute in 2000. The exact description of the problem is given by Charles L. Fefferman [18] in the short article "Existence and Smoothness of the Navier-Stokes Equations". To be more precise, following the formulation given by Charles L. Fefferman we recall the exact description of the problem, which is restricted to incompressible fluids. The Navier-Stokes equations are then given by

$$\frac{\partial}{\partial t}u_i + \sum_{j=1}^{n} u_j \frac{\partial u_i}{\partial x_j} - \nu \Delta u_i + \frac{\partial p}{\partial x_i} = f_i(x,t), \qquad x \in \mathbb{R}^n, t \geq 0, \tag{1.1}$$

$$\operatorname{div}(u) = 0, \qquad x \in \mathbb{R}^n, t \geq 0, \tag{1.2}$$

with initial conditions

$$u(x,0) = u^0(x), \qquad x \in \mathbb{R}^n, \tag{1.3}$$

where u^0 is a given, C^∞ divergence-free vector field on $\mathbb{R}^n$, f_i are the components of a given function, ν is a positive coefficient and Δ describes the Laplace operator. A solution is considered being physical only if it satisfies

$$p, u \in C^\infty\left(\mathbb{R}^n \times [0,\infty)\right), \tag{1.4}$$

$$\int_{\mathbb{R}^n} |u(x,t)|^2 \, dx < C \qquad \text{for all} \qquad t > 0. \tag{1.5}$$

Existence and smoothness of solutions to the Navier-Stokes equations on $\mathbb{R}^n$
Assume that $n = 3$, $\nu > 0$, $f = 0$ and that u^0 is any smooth, divergence-free vector field satisfying

$$\left|\partial_x^\alpha u^0(x)\right| \leq C_{\alpha K}\left(1 + |x|\right)^{-K}, \qquad x \in \mathbb{R}^n, \quad \alpha, K \geq 0. \tag{1.6}$$

Then there exist smooth functions p, u_i on $\mathbb{R}^n \times [0,\infty)$ that satisfy (1.1), (1.2), (1.3), (1.4) and (1.5).

Breakdown of solutions to the Navier-Stokes equations on $\mathbb{R}^n$
Assume that $n = 3$,and $\nu > 0$. Then there exist a smooth, divergence-free vector field u^0 on $\mathbb{R}^3$ and a smooth function f on $\mathbb{R}^3 \times [0,\infty)$ satisfying (1.6) and

$$|\partial_x^\alpha \partial_t^m f(x,t)| \leq C_{\alpha m K}\left(1 + |x| + t\right)^{-K}, \qquad x \in \mathbb{R}^n \times [0,\infty), \quad \alpha, m, K \geq 0,$$

for which there exist no solutions u and p of (1.1), (1.2), (1.3), (1.4) and (1.5) on $\mathbb{R}^3 \times [0,\infty)$.

The Millennium Prize Problem is also formulated for periodic problems. These problems are also open for the Euler equations, though they are not on the Clay Institute's list of prize problems. We close this short discussion by the final remark of Charles L. Fefferman: "Fluids are important and hard to understand. There are many fascinating problems and conjectures about the behavior of solutions of the Euler and Navier-Stokes equations. (See, for instance, Bertozzi-Majda [5] or Constantin [12]). Since we don't even know whether these solutions exist, our understanding is at a very primitive level."

Within this thesis we do not neglect compressibility effects. The exact form of the equations considered is stated in Chapter 2. The incompressible Navier-Stokes equations (1.1) and (1.2) formulated above are a specialization of the one considered in this thesis. Since the existence of solutions to the incompressible Navier-Stokes equations is unknown, it is also unknown for the governing equations formulated in Chapter 2. Hence, it is the topic of this thesis to suggest, investigate and implement methods to approximate solutions to a set of equations for which even the existence

of solutions is not settled. This fact may question the motivation of this whole work. Critical readers may stop at this point to ask for a settled theoretical background.

On the other hand, considering the importance of the understanding of solutions to these equations, the implementation and the finding of approximate solutions are of major importance for many practical problems. Nevertheless, the author of this thesis is convinced that as long as the theoretical background is missing and the understanding of these equations "is at a very primitive level", the implementations of algorithms which try to approximately solve these equations can only be of limited success. It is the theoretical, mathematical background enforcing the rules to solve these equations.

Though there is a great lack of theoretical understanding, due to their importance for many practical engineering problems, and the significant increase in computational power over the last decades, an immense number of computer codes have been implemented to approximately solve the Euler, the Navier-Stokes or the RANS equations. More exact, these equations are used to formulate the corresponding boundary value problems which are then approximately solved. In general, the domain of interest is approximated by a grid, and the differential or integral operators are approximated using geometrical data generated from the grid. Going hand in hand with this, a manner of speaking has been established categorizing these computer codes into two different classes:

a) Structured grid codes,

b) mixed element grid codes.

For structured grid codes the domain of interest is assembled using only quadrilateral elements in 2D and hexahedral elements in 3D. Each element can be addressed directly. In structured grid codes this direct addressing is exploited in a straightforward manner to realize efficient implementations of algorithms. For example, considering standard finite difference stencils the residual for some element in a given structured mesh can be directly computed. Though the efficiency and use of structured grid codes, due to their simplicity, is advantageous, it turns out that automatic mesh generation is often extremely difficult for complex geometries, such as a complete helicopter or the gear of an aircraft. Industrial requirements such as automatism pose a significant challenge for routine simulations for complex configurations on structured or block-structured meshes. A more realistic situation is that a computational mesh is generated with an arbitrary, fully automatic mesh generator.

Hence, meshing strategies which allow the use of additional elements such as triangles, tetrahedra, pyramids and prisms are established. Historically, it seems that first the quadrilateral and hexahedral element meshes were replaced by triangular

and tetrahedral meshes. Using only these elements for unstructured grids, unsatisfactory accuracy was observed for high Reynolds number turbulent flows. For example, with comparable degrees of freedom, the accuracy obtained with structured grids could often not be reproduced with such unstructured grids. As a necessity, it became clear that in regions of the mesh where steep gradients need to be resolved (such as the boundary layer around a body) that quadrilateral and hexahedral elements are required. Then discretization schemes comparable to structured grid codes are obtained. To combine the hexahedral elements with tetrahedrons, additional elements such as prisms and pyramids were introduced, resulting in mixed element meshes.

These mixed element meshes only allow for an indirect addressing of the elements. The additional exploited information from structured grid codes is lost. Therefore, well known techniques applied in structured grid codes need to be generalized to mixed element grid codes. As an example, we mention the multigrid technique used for acceleration of solution algorithms. This technique is straightforward to implement in structured grid codes exploiting the direct addressing of the elements by fusing neighboring elements. Within a mixed element grid, this direct information is missing. One way of generalization of the original idea is the representation of the mesh and its connectivities by a graph, and then the aggregation of elements is done by the identification of suited subgraphs. More general, there is a major consequence of this fully automatic mesh generation idea; namely, it excludes that the solution method is allowed to make detailed assumptions about the given grid. This makes the problem to find efficiently an accurate solution on a given grid inherently more complicated.

Besides classical finite volume discretization schemes over the last decade, so-called high order methods such as discontinuous Galerkin (DG) have been established. In contrast to classical finite elements methods the DG methods neglect the construction of global continuous ansatz functions. Instead one uses polynomials of certain degree only inside one element. The connection to the surrounding elements is realized using the same flux functions originally derived for finite volume methods. Then, the finite volume method is interpreted as a DG method with piecewise constant ansatz functions. In consequence, these methods represent a natural generalization of the finite volume schemes. Besides the immense increase in complexity, one major challenge of these methods is the requirement for a high order representation of the computational mesh.
To be more precise, the surfaces of each element need to be represented by a polynomial. To reach the design order of the method the quadrature formulae and quadrature points, often a Gauss-Legendre or Gauss-Lobatto method, need to be adapted to the polynomial representation of the elements. The difficulty to generate valid meshes for complex geometries comes from the challenge to generalize the requirement for positive volumes to orientable manifolds. Further challenges for discontinuous Galerkin discretizations are for example the resolution of shocks. Due

to these difficulties as well as the fact that analytic solutions of the compressible Navier-Stokes equations are possibly non-smooth, the advantage to approximate these solutions by polynomials of high order is in general hard to assess. Over the last years it seems that these methods have been established in the field of scale resolving simulations. In this research area the considered geometries are often rather simple and can be represented by structured grids. Though high order methods are mathematically attractive, it seems that the full potential and the field of application is not yet clarified.

The different ways to represent grids go hand in hand with the discretization of the set of equations are basically a technical question. It is worthwhile to note that from an industrial point of view the meshing strategies are a major contributing factor. Mathematically and algorithmically the original system of equations inherits its properties to the discretized system of equations. With respect to this point of view, essentially the Navier-Stokes equations represent a time dependent set of partial differential or integral equations. Given an initial condition at time $t = 0$ it is of interest to find an integral curve satisfying the corresponding initial boundary value problem.

Throughout this thesis it is not the goal to approximate the time envelope of such an integral curve. We assume that after a certain time $T > 0$ a solution to the equations is not time dependent any more. Hence we are only interested in a steady state solution. This problem can be reformulated as an operator equation

$$F(x) = y, \tag{1.7}$$

where $F : D(F) \to \mathcal{Y}$ is a nonlinear operator between its domain $D(F) \subset \mathcal{X}$ into $\mathcal{Y}$. For a correct and complete mathematical setting, it is required to characterize F, $\mathcal{X}$ and $\mathcal{Y}$. Since mathematical theory is missing, we assume that F is an injective Fréchet differentiable mapping between the Hilbert or Banach spaces $\mathcal{X}$ and $\mathcal{Y}$. We leave it open to specify the scalar products or norms which define the metrics for $\mathcal{X}$ and $\mathcal{Y}$ such that (1.7) is well posed. This information is not at hand. Moreover, the assumptions made about F are also possibly incorrect.

Nevertheless, the assumptions made, allow applying Newton's method. It is one of the most powerful techniques for solving nonlinear equations. Its widespread applications in all areas of mathematics make it one of the most important and best known procedures in this science. Usually it is the first choice to try for solving some given nonlinear equation. Many other good methods designed to solve nonlinear equations often turn out to be variants of Newton's method attempting to preserve its convergence properties without its disadvantages. A motivation of Newton's method is given by the following elementary construction:

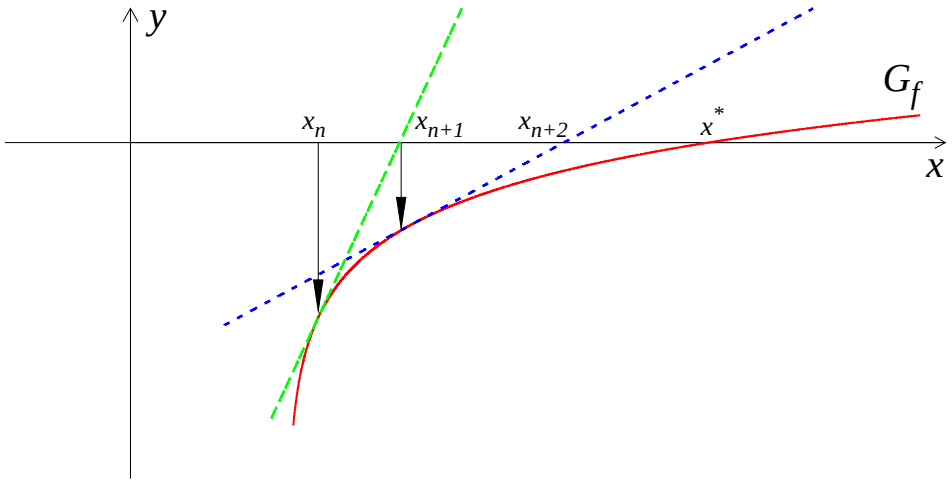

Figure 1.1: An illustration of Newton's method

We consider the nonlinear equation $f(x) = 0$, where $f : \mathbb{R} \to \mathbb{R}$ is a continuously differentiable function. Let x_n be an approximation to some root x^* of f and $y(x) := f(x_n)+f'(x_n)(x-x_n)$ the tangent on f through $(x_n, f(x_n))$. If $f'(x_n) \neq 0$, then y has exactly one point of intersection with the x-axis, which we examine as new approximation to x^*. Proceeding in this way, which is illustrated in Figure 1.1, we obtain the algorithm

$$x_{n+1} := x_n - [f'(x_n)]^{-1} f(x_n), \qquad n = 0, 1, 2, \ldots.$$

This idea can be generalized to the operator equation (1.7). Substituting F by its linear approximation in each Newton step, the least squares problem

$$\|F'(x_n)h + F(x_n) - y\|_{\mathcal{Y}}^2 = \min_{h \in \mathcal{X}} ! \tag{1.8}$$

needs to be solved. $F'(x_n)$ denotes the Fréchet derivative of F at x_n and the Newton update is given by $h = x_{n+1} - x_n$. This generalized approach is well-known as Gauss-Newton method. If the operator equation (1.7) is well posed, many different local convergence proofs of the Gauss-Newton method have been established to show convergence of quadratic order under some natural conditions on the operator F. Unfortunately, a straightforward implementation of this solution method to approximate a steady state solution of a boundary value problem corresponding to the Navier-Stokes equations is in general not successful. Additional ingredients and specification of this method are required.

As a further powerful tool to solve linear and nonlinear equations, multigrid methods are established. It is the basic idea of multigrid methods to combine a cheap,

iterative solution method with a hierarchy of operators such that the original problem can be solved efficiently. Originally these methods were designed for linear equations. One famous example arises from the discretization of Poisson's equation on a unit square. Considering a Cartesian grid, which is coarsened equally, one obtains for each successive coarser grid level a linear equation which approximates the equation on the next finer grid level. Then, using for example a Jacobi or Gauss-Seidel method, to smooth the solution on each grid level together with an exact solver on the coarsest level one obtains a powerful solution method. It can be proven that this method requires $O(N)$ operations; that is, it scales only linearly with the number of degrees of freedom.

For nonlinear problems the multigrid idea has been generalized by what is called the **F**ull **a**pproximation **s**torage (FAS) multigrid method. The FAS multigrid has been developed and applied to solve problems arising from boundary value problems corresponding to the Euler and Navier-Stokes equations. To go along with the original idea of multigrid, it has been combined with cheap smoothers given by explicit multistage Runge-Kutta methods, which indeed have a very low operation count per iteration. These methods found their way into many codes. However for many applications, it is observed that even the introduction of multigrid did not overcome the problems inherent to solve the Navier-Stokes equations as long as such weak smoothers were used.

A major factor contributing to the loss in effectiveness of solution methods is that anisotropic meshes are generally used for the economic resolution of the steep gradients occurring in viscous boundary layers. Such meshes have high aspect ratio cells, resulting in a stiffness of the discrete system of governing flow equations. It is this numerical stiffness that creates difficulty in removing certain error modes when computing flow solutions. For analysis of the effects of such stiffness see Pierce and Giles [73]. These results questioned the basic idea, if for these kinds of problems the combination of multigrid with a cheap smoother can be a successful way. This question is even more severe when one considers the simulation of complex flows which in general require a significant increase in the number of degrees of freedom needed to resolve the main flow features.

Then, about a decade ago, it was shown by Rossow, Swanson and Turkel [80, 93, 91] that a significant improvement in both reliability and efficiency can be reached when the explicit Runge-Kutta smoother is supplemented by an implicit one. This observation was true not only for one single test case but for several examples. In [93] the efficiency of the scheme was demonstrated for a 3D wing flow using the Baldwin-Lomax algebraic turbulence model [3]. In [9] a similar method was introduced into a block-structured code and the superiority of the method when compared with explicit or point-Jacobi preconditioned [109, 42] Runge-Kutta methods was also shown for 3D flows. However, in [9] convergence of the turbulent flow equation given by an SA model was not shown. Mainly all results shown were

restricted to structured grid codes, and how to carry these methods over to mixed element codes was not clear.

Nevertheless, these results suggested that to overcome the stiffness of problems one has to combine the FAS multigrid method with a powerful method to solve nonlinear equations, namely kinds of Newton's method. This observation contradicts the basic idea of multigrid, but it accommodates the fact that solving the Navier-Stokes equations themselves are a challenge. Further, in combination with high Reynolds-numbers, additional equations and variables due to the usage of turbulence models and meshes with significant anisotropies are an even greater challenge. It is not a matter of course to solve these sets of equations.

The challenge in solving these equations is evident because even the combination of FAS multigrid and Newton's method as a smoother are in general not able to approximate a solution unless some globalization strategy is introduced. This observation is reasonable, since for large scale, complex configurations a good initial guess is often not available. So, strategies are required and need to be included into the algorithm which overcome the start-up problem. Mathematically and algorithmically, none such globalization methods are known which work in the generality which is necessary. The design of these strategies is open. Within this thesis the main globalization strategy considered is the generalization of Newton's method to some special kind of multistage, implicit Runge-Kutta method. One can also interpret this method as a regularization strategy. This approach helps for many problems, but it should not over-estimated. The second major problem with a direct implementation of Newton's method is to approximate a solution of the inner linear systems. It turns out that for the problems considered in this thesis, these linear systems are ill-conditioned. Hence, stabilization strategies are required to approximate solution.

To summarize the discussion above we are interested in four aspects, which are of particular importance in the investigation of solution methods for the Navier-Stokes equations.

a) **Accuracy**: Starting from a full description of the discretization strategy. Are there methods and possibilities to evaluate the accuracy of the approximate solution?

b) **Combination of multigrid with implicit smoother**: What are the main ingredients to construct a reliable algorithm to find approximate solutions of the RANS equations for high Reynolds number turbulent flows?

c) **Application to the incompressible limit**: Are there possibilities to extend the discretization and solution strategy to the incompressible limit to get a closed formulation for the simulation of incompressible and compressible flows, that is to solve the compressible equations for low inflow Mach numbers?

d) **Analysis to assess the methods**: Besides heuristic arguments, are there possibilities to assess the developed methods?

This thesis is roughly divided into four parts. Chapter 2 presents the governing equations of fluid dynamics. The equations include the Euler and Navier-Stokes equations as well as the turbulent flow equations to determine the eddy viscosity required as closure condition for the RANS equations. The discretization strategy and the total derivatives are carried out in Chapters 3 and 4. The third part of the thesis is dedicated to describing the construction of solution algorithms. Details about this topic are given in Chapter 5. Finally, Chapters 6 and 7 are devoted to numerical examples and scalability investigations necessary to evaluate the algorithms with respect to their parallelization capabilities. We close the introduction by a detailed explanation concerning the topic of each chapter.

In **Chapter 2** we present the governing equations. This presentation comprises both the RANS equations as well as two kinds of established turbulence models required to determine the eddy viscosity. Based upon the author's experience, one rarely finds a complete summary. Here the nondimensionalization of all these equations is also given. For completeness, corresponding boundary value problems including the Euler- and the Navier-Stokes equations are formulated. Here, also the analytic boundary conditions for the turbulent flow equations are included. Since turbulence modeling is important in the field of computational fluid dynamics, the Chapter is closed by a mathematical point of view of this complex topic.

Chapter 3 is dedicated to the discretization strategy. The basic discretization strategy considered is a finite volume method for mixed element grids. Within this thesis we derive this method by a discontinuous Galerkin method using constant ansatz functions. This approach gives evidence to the fact that the established finite volume methods can be interpreted as a specialization of this more general class of discretization strategy. To discretize the convective contribution, a numerical flux function based on a central difference scheme is presented. Two different methods to extend the scheme for low Mach number flows are derived. The extension of an existing scheme originally implemented for compressible flows to the incompressible limit is a long term scientific topic in the world of computational fluid dynamics. For the discretization of the viscous contribution, both a Green-Gauss method to approximate the gradients is used and a thin shear layer assumption are implemented. Theoretical facts as well as numerical examples are used to investigate the effect of these two different discretization schemes. The implementation of all boundary conditions is based on a flux formulation. A complete description of the realization of the boundary conditions is given. Moreover, throughout the literature one rarely finds a complete presentation of the implementation of the additional turbulence flow equations. Therefore, this topic is also treated with care.

In **Chapter 4** the derivatives required to formulate an implicit solution method are presented. It was observed in many numerical examples that it is often very impor-

tant for a successful application of an implicit solution method that the derivatives are based on the actual discretization strategy. In particular, the inclusion of the derivatives of the boundary conditions was figured out to be of major importance. Hence, great importance is attached to a detailed presentation of the derivatives such that in future work based upon these results easily additional derivatives can be included. Note, the focus here is not only the mathematical realization of each of the terms but also the construction of a half automatic differentiation method such that in future work a good framework is given to add further terms. Also, the construction and realization of the data structures to save and to work with large scale block sparse matrices is presented.

Chapter 5 suggests an approach for constructing general, powerful solution algorithms to approximate solutions of nonlinear equations. In particular, it will be shown that almost all solution algorithms suggested throughout the literature of computational fluid dynamics are specializations of the general framework suggested in this thesis. This knowledge puts us into a position to assess and classify the several solution methods. The driver of the solution algorithm is an FAS multigrid scheme with an implicit smoother. The smoother is derived by a multistage diagonal implicit Runge-Kutta method. Being only interested in steady state solutions, the global time step is replaced by a local one for convergence acceleration. It is concluded that the multistage ansatz as well as the local time step are only methods to stabilize Newton's method. And indeed, it will be shown that for certain parameter choices the smoother turns out to be Newton's method. Hence, the suggested algorithm is simply some kind of regularized Newton's method, which is close to a Levenberg-Marquardt algorithm. Analysis tools are designed giving evidence that these kinds of regularization and globalization strategies are required to construct stable algorithms to approximate solutions of the Navier-Stokes equations. Furthermore, it will be shown that simplifications yield to unstable methods.
To construct and try several algorithms, the main features are understood as building blocks. To realize and to allow for an easy exchange of components a modular software design was aspired. This put us into a position that the inner linear systems arising in the implicit Runge-Kutta method can be approximately solved by a whole variety of methods staring with simple Gauss-Seidel type algorithms and including linear multigrid methods. These methods in general serve as preconditioners for an outer Krylov solver, typically GMRES. Further details such as the construction of agglomerated meshes can be found in this Chapter 5.

In **Chapter 6** we address applications side. Several examples will be considered to investigate and evaluate both the discretization strategies as well as the corresponding solution algorithms. First, the focus is the assessment of the solution algorithms. For several test cases different solution algorithms are tested and compared to give an idea about their potential and shortcomings. It is not the goal of this thesis to identify one certain algorithm and to show its superiority compared to all others. The author of this thesis is also convinced that in general such an assertion is not

possible due to the variety of parameters and several possibilities to combine the building blocks. Here, the idea is to show the effect of certain key components and their necessity. Second, our interest is also with respect to the accuracy of the scheme. This is an important topic, and for this topic as well, no clear answer in general can be expected. It is not only the discretization strategy which guides the accuracy, but also the mesh and naturally the smoothness of an analytic solution. In particular, smoothness of an analytic solution is on the one hand unknown, nor can it be expected. For example, flows with a shock, stagnation points, boundary conditions, corners and edges of the geometry and many other reasons can be considered to conclude that already an analytic solution is non-smooth. In absence of this property, the expectation of the approximate solution converging on a sequence of refined meshes of more than order one to an analytic solution cannot be satisfied in general.

In **Chapter 7** parallelization issues are briefly investigated. It is one of the most important demands nowadays that suggested and implemented algorithms work efficiently on high performance computing clustering systems. Such an investigation shows to what extent the additional employed resources pay off. In general, there is a distinction between strong scaling and weak scaling of algorithms. Roughly speaking, the considered problem size for the latter one is increased when one uses more resources. Hence, to show that an algorithm scales well in a weak sense is often not too difficult. Therefore, an assertion about weak scaling is often of minor interest. Hence, the investigation considered here is with respect to strong scaling. Here the problem size is fixed and the resources to solve the problem are systematically increased. Due to the fact that each subproblem is getting smaller and smaller, it is expected that from some point the additional employed resources will not pay off any more. The investigation, when this is going to happen, is considered in this chapter.

Finally, we discuss our results and conclude this thesis with an outlook in **Chapter 8**.

Chapter 2

The RANS equations

In this chapter we present the Reynolds averaged Navier-Stokes (RANS) equations considering Boussinesq' eddy viscosity assumption. To determine the a-priori unknown eddy viscosity, two in the field of computational fluid dynamics established turbulence models are introduced. These are called the **S**palart-**A**llmaras (SA) model and the $k\omega$-model of Wilcox. Starting from the RANS equations the mathematically connection to the Navier-Stokes equations and the Euler equations is explained and corresponding boundary value problems are formulated. Since a complete presentation of nondimensionalization of the equations could not be found in the literature, in particular not of the additional turbulence flow equations, we have emphasized this topic in the second part of this chapter. Due to the importance of turbulence modeling, which is required to resolve the main characteristics of high Reynolds number turbulent flows, this chapter closes with a section about a mathematical view on this complex topic.

2.1 Governing equations

In this chapter we deal with both the dimensional and the nondimensional form of the Navier-Stokes equations. In all the following chapters only the nondimensional form is of interest. To distinguish between dimensional and nondimensional variables, the dimensional quantities are marked with a $\hat{}$. For example, the dimensional physical time is denoted by $\hat{T}$ and the physical density by $\hat{\rho}$. Section 2.3 is devoted to carry the dimensional form of the equations over to their nondimensional form, which is then the only relevant form of the equations considered throughout this thesis.

To describe flow effects we consider for the domain $\hat{D} \subset \mathbb{R}^m$, $m = 2, 3$, i.e., an open and connected set, and an interval $[0, \hat{T}) \subset \mathbb{R}$, $\hat{T} > 0$, the RANS equations in conservative form. These are a system of non-linear conservation laws which results naturally from the fundamental laws of conversation of mass, momentum

and energy. The dimensional form of the governing equations can be expressed in integral form by

$$0 = \frac{d}{d\hat{t}} V_{\hat{D}}\left(\hat{W}\right)(\hat{t}) + R_{\partial\hat{D}}\left(\hat{W}\right)(\hat{t}), \qquad \hat{t} \in \left[0, \hat{T}\right), \tag{2.1a}$$

where the integral operators $V_{\hat{D}}$ and $R_{\partial\hat{D}}$ are given by

$$V_{\hat{D}}\left(\hat{W}\right)(\hat{t}) := \int_{\hat{D}} \hat{W}(\hat{x}, \hat{t})\, d\hat{x} \tag{2.1b}$$

$$R_{c,\partial\hat{D}}\left(\hat{W}\right)(\hat{t}) := \int_{\partial\hat{D}} \left\langle f_c\left(\hat{W}(\hat{y}, \hat{t})\right), n(\hat{y})\right\rangle ds(\hat{y}), \tag{2.1c}$$

$$R_{v,\partial\hat{D}}\left(\hat{W}\right)(\hat{t}) := \int_{\partial\hat{D}} \left\langle f_v\left(\hat{W}(\hat{y}, \hat{t})\right), n(\hat{y})\right\rangle ds(\hat{y}), \tag{2.1d}$$

$$R_{\partial\hat{D}} := \hat{R}_{c,\partial\hat{D}} - \hat{R}_{v,\partial\hat{D}}, \tag{2.1e}$$

and $\hat{W} : \hat{D} \times [0, \hat{T}) \to \mathbb{R}^{m+2}$,

$$\hat{W}(\hat{x}, \hat{t}) := \left(\hat{\rho}(\hat{x}, \hat{t}), \hat{\rho}(\hat{x}, \hat{t})\hat{u}(\hat{x}, \hat{t}), \hat{\rho}(\hat{x}, \hat{t})\hat{E}(\hat{x}, \hat{t})\right)^T,$$

denotes the vector field of conserved variables and n is the unit outward normal on $\partial\hat{D}$. The terms f_c and f_v describe the convective and viscous contribution

$$f_c\left(\hat{W}\right) := \begin{pmatrix} \hat{\rho}\hat{u} \\ \hat{\rho}\hat{u}_1\hat{u} + \hat{p}e_1 \\ \vdots \\ \hat{\rho}\hat{u}_m\hat{u} + \hat{p}e_m \\ \hat{\rho}\hat{H}\hat{u} \end{pmatrix}, \qquad f_v\left(\hat{W}\right) := \begin{pmatrix} 0 \\ \tau_1\left(\hat{W}\right) \\ \vdots \\ \tau_m\left(\hat{W}\right) \\ \theta\left(\hat{W}\right) \end{pmatrix}, \qquad m = 2, 3,$$

and $\langle f_c, n\rangle$ and $\langle f_v, n\rangle$ are called the convective and viscous flux in normal direction n. The expression

$$\langle x, y\rangle := \sum_{j=1}^{m} x_j y_j, \qquad x, y \in \mathbb{R}^m,$$

denotes the standard l^2 scalar product in $\mathbb{R}^m$ and needs to be understood component by component for each of the equations in (2.1a). Throughout this thesis the symbols $e_1, \ldots, e_m$ denote the standard orthonormal basis of $\mathbb{R}^m$, i.e.

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, e_m = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

The dimensional quantities $\hat{\rho}$, $\hat{u} = (\hat{u}_1, \ldots, \hat{u}_m)^T$, $\hat{E}$ and

$$\hat{H} := \hat{E} + \hat{p}/\hat{\rho} \tag{2.2}$$

are the density, velocity, the specific total energy, and the enthalpy of the fluid. The momentum in $\hat{x}_j$ direction is given by the product of density and velocity, $\widehat{mom}_j := \hat{\rho}\hat{u}_j$. The equation of state

$$\hat{p}\left(\hat{W}\left(\hat{x},\hat{t}\right)\right) := (\gamma-1)\hat{\rho}\left(\hat{x},\hat{t}\right)\left(\hat{E}\left(\hat{x},\hat{t}\right) - \frac{\|\hat{u}\left(\hat{x},\hat{t}\right)\|_2^2}{2}\right) \tag{2.3}$$

defines the pressure $\hat{p}$ and γ is the gas dependent ratio of specific heats, which is given by 1.4 for air. The speed of sound $\hat{a}$, dimensionless Mach number M and temperature $\hat{T}$ are defined by

$$\hat{a} := \sqrt{\frac{\gamma\hat{p}}{\hat{\rho}}}, \qquad M := \frac{\|\hat{u}\|_2}{\hat{a}}, \qquad \hat{T} := \frac{\hat{p}}{\hat{\rho}\hat{\Re}}, \tag{2.4}$$

where $\hat{\Re}$ describes the universal gas constant. The speed of sound $\hat{a}$ can be reformulated using the equation of state (2.3) and (2.2)

$$\begin{aligned}
\hat{a}^2 &= \frac{1}{\hat{\rho}}\left[\hat{p} + (\gamma-1)\hat{p}\right] \\
&= \frac{1}{\hat{\rho}}\left[(\gamma-1)\hat{\rho}\left(\hat{E} - \frac{\|\hat{u}\|_2^2}{2}\right) + (\gamma-1)\hat{p}\right] \\
&= (\gamma-1)\left(\hat{E} + \frac{\hat{p}}{\hat{\rho}} - \frac{\|\hat{u}\|_2^2}{2}\right) \\
&= (\gamma-1)\left(\hat{H} - \frac{\|\hat{u}\|_2^2}{2}\right).
\end{aligned} \tag{2.5}$$

To give the definition of the viscous flux f_v we introduce the strain rate tensor.

Definition 2.1.1 *The strain rate tensor $\mathcal{S} = \mathcal{S}(\hat{u}) = \mathcal{S}\left(\hat{u}\left(\hat{x},\hat{t}\right)\right)$ is given by the symmetric part of the total derivative of the flow velocity $\hat{u}$,*

$$\mathcal{S}(\hat{u}) := \frac{1}{2}\left(\frac{d\hat{u}}{d\hat{x}} + \left(\frac{d\hat{u}}{d\hat{x}}\right)^T\right).$$

The trace free shear stress tensor $\overline{\mathcal{S}}$ is denoted by

$$\overline{\mathcal{S}}(\hat{u}) := \mathcal{S}(\hat{u}) - \frac{1}{3}\operatorname{div}(\hat{u})\, Id. \tag{2.6}$$

Additionally, we define the vorticity $\Omega = \Omega(\hat{u}) = \Omega\left(\hat{u}\left(\hat{x},\hat{t}\right)\right)$ by the skew-symmetric part of the total derivative of the flow velocity $\hat{u}$,

$$\Omega(\hat{u}) := \frac{1}{2}\left(\frac{d\hat{u}}{d\hat{x}} - \left(\frac{d\hat{u}}{d\hat{x}}\right)^T\right).$$

Corollary 2.1.2 *For $m = 3$ The trace free shear stress tensor $\overline{\mathcal{S}}$ is trace free.*

Proof: The assertion follows for $m = 3$ by

$$\mathrm{tr}\left(\overline{\mathcal{S}}\right) = \mathrm{tr}\left(\mathcal{S}\right) - \frac{1}{3}\mathrm{tr}\left(\ \mathrm{div}\left(\hat{u}\right) Id\right) = \ \mathrm{div}\left(\hat{u}\right) - \frac{1}{3}\left(m\ \mathrm{div}\left(\hat{u}\right)\right) = 0.$$

□

Assuming that an effective viscosity

$$\mu_{\mathrm{eff}} := \mu_{\mathrm{eff}}\left(\hat{W}\right) = \mu_{\mathrm{eff}}\left(\hat{W}\left(\hat{x},\hat{t}\right)\right)$$

is given and, using Stoke's hypothesis, that the bulk viscosity satisfies $\lambda = -2/3\mu_{\mathrm{eff}}$, the viscous stress tensor $\tau = \tau\left(\hat{W}\right) = \tau\left(\hat{W}\left(\hat{x},\hat{t}\right)\right)$ is given by

$$\tau\left(\hat{W}\right) := \mu_{\mathrm{eff}}\mathcal{S} + \lambda\,\mathrm{div}(\hat{u})Id = 2\mu_{\mathrm{eff}}\left(\mathcal{S} - \frac{1}{3}\,\mathrm{div}(\hat{u})Id\right) = 2\mu_{\mathrm{eff}}\overline{\mathcal{S}}. \tag{2.7}$$

Hence, τ is symmetric and can be explicitly expressed by

$$\begin{aligned}
\tau_{ii}\left(\hat{W}\right) &= 2\mu_{\mathrm{eff}}\frac{\partial\hat{u}_i}{\partial\hat{x}_i} + \lambda\,\mathrm{div}(\hat{u}) = \frac{2}{3}\mu_{\mathrm{eff}}\left(2\frac{\partial\hat{u}_i}{\partial\hat{x}_i} - \sum_{j=1,j\neq i}^{m}\frac{\partial\hat{u}_j}{\partial\hat{x}_j}\right), \qquad i = 1,\ldots,m,\\
\tau_{ij}\left(\hat{W}\right) &= 2\mu_{\mathrm{eff}}\mathcal{S}_{ij} = \mu_{\mathrm{eff}}\left(\frac{\partial\hat{u}_i}{\partial\hat{x}_j} + \frac{\partial\hat{u}_j}{\partial\hat{x}_i}\right), \qquad \tau_{ji} = \tau_{ij}, \qquad 1 \le i < j \le m.
\end{aligned}$$

The missing viscous flux term for the energy equation is given by

$$\theta\left(\hat{W}\right) := \tau\left(\hat{W}\right)\hat{u} + q\left(\hat{W}\right), \tag{2.8a}$$

$$q\left(\hat{W}\right) := \kappa_{\mathrm{eff}}\ \mathrm{grad}\ \hat{T}, \tag{2.8b}$$

which can be expressed componentwise by

$$\theta_j\left(\hat{W}\right) := \left(\sum_{k=1}^{m}\tau_{jk}\left(\hat{W}\right)\hat{u}_k\right) + \kappa_{\mathrm{eff}}\frac{\partial\hat{T}}{\partial\hat{x}_j}, \qquad j = 1,\ldots,m.$$

The effective viscosity μ_{eff} and effective conductivity κ_{eff} are computed by

$$\mu_{\mathrm{eff}} := \mu_l + \mu_t, \qquad \kappa_{\mathrm{eff}} := \kappa_l + \kappa_t, \tag{2.9}$$

and the laminar viscosity is given by Sutherland's law

$$\mu_l\left(\hat{W}\right) := \hat{\mu}_{l,\infty}\left(\frac{\hat{T}}{\hat{T}_\infty}\right)^{3/2}\frac{\hat{T}_\infty + \hat{\tilde{T}}}{\hat{T} + \hat{\tilde{T}}}, \qquad \hat{\mu}_{l,\infty} := \frac{\hat{\rho}_\infty\hat{u}_\infty\hat{L}}{Re}, \tag{2.10}$$

$$\kappa_l\left(\hat{W}\right) := \frac{\hat{c}_p\mu_l\left(\hat{W}\right)}{Pr_l} \qquad \text{and} \qquad \hat{c}_p := \hat{\Re}\frac{\gamma}{\gamma-1}, \tag{2.11}$$

whereby $\hat{\rho}_\infty > 0$ and $\hat{u}_\infty > 0$ denote some constant reference density and velocity, $\hat{L} > 0$ is some constant reference length scale, $Re > 0$ is the corresponding Reynolds number,

$$\hat{\tilde{T}} := 110.4\text{K} \tag{2.12}$$

is Sutherland's constant, $\hat{\Re}$ is the universal gas constant and the laminar Prandtl number is given by $Pr_l := 0.72$. The choice of the not yet determined constant values is topic of Section 2.3. For later use, according to the laminar dynamic viscosity μ_l we define the laminar kinematic viscosity by

$$\nu_l\left(\hat{W}\right) := \frac{\mu_l\left(\hat{W}\right)}{\hat{\rho}}. \tag{2.13}$$

Finally, computational prescriptions of the additional unknowns introduced in (2.9), namely the eddy viscosity μ_t and the turbulent thermal conductivity κ_t are required. Given the eddy viscosity μ_t the turbulent thermal conductivity is described by the algebraic relation

$$\kappa_t := \hat{c}_p \frac{\mu_t}{Pr_t}, \qquad Pr_t := 0.92. \tag{2.14}$$

Here the turbulent Prandtl number Pr_t was introduced. In all the examples considered in Chapter 6 we chose for the turbulent Prandtl number Pr_t the value given in (2.14). Note that one also finds values of $Pr_t = 0.9$ [17, p. 13] and $Pr_t = 0.91$ in the literature. The algebraic relation (2.14) reduces the two additional unknowns to one.

To define the eddy viscosity μ_t is not straightforward. It is the topic of turbulence modeling to provide a physically reasonable function μ_t to simulate turbulent fluid flows. In our context it is computed from additional unknowns denoted in the following by $\hat{W}_t$. These are the solution of additional equations, so-called turbulence flow equations, which will be presented in Section 2.2 below. Here, at the outset we postulate that

$$\mu_t\left(\hat{W}_t, \hat{W}\right)(\hat{x}, \hat{t}) \geq 0 \qquad \text{for all} \qquad \left(\hat{x}, \hat{t}\right) \in \hat{D} \times \left[0, \hat{T}\right). \tag{2.15}$$

Note, the formulation of the RANS equations (2.1a) allows for any reasonable function μ_t. It does not necessarily need to be determined by further complex systems of differential or integral equations. Assuming this function is known a-priori, it could be simply fed into the set of equations (2.1a). Finally, corresponding to the laminar kinematic viscosity (2.13) we define the turbulent kinematic viscosity by

$$\nu_t\left(\hat{W}_t, \hat{W}\right) := \frac{\mu_t\left(\hat{W}_t, \hat{W}\right)}{\hat{\rho}}. \tag{2.16}$$

Before we introduce the turbulence models of Spalart and Allmaras [85, 1] and the $k\omega$-model of Wilcox [112, 113], we give a short definition of the set of equations of interest in this thesis.

Definition 2.1.3 *Consider the system of integral equations (2.1a).*

a) Assume that $f_v \equiv 0$ *for all* $\left(\hat{x}, \hat{t}\right) \in \hat{D} \times \left[0, \hat{T}\right)$. *Then we call (2.1a) the Euler equations, that is (2.1a) is reduced to*

$$0 = \frac{d}{d\hat{t}} V_{\hat{D}}\left(\hat{t}\right) + R_{c,\partial\hat{D}}\left(\hat{W}\right)\left(\hat{t}\right), \qquad \hat{t} \in \left[0, \hat{T}\right).$$

b) Assume that $\mu_t \equiv \kappa_t \equiv 0$. *Then we call (2.1a) the (laminar) Navier-Stokes equations.*

c) Otherwise we call (2.1a) the Reynolds-averaged Navier-Stokes equations.

Remark 2.1.4 *To obtain from the system of integral equations (2.1a) the Euler equations it is equivalent to assume that* $\mu_t \equiv \mu_l \equiv 0$. *Hence, the Euler equations neglect diffusive and turbulence effects. That is, these equations model inviscid flows.*

2.2 Turbulence models

Turbulence modeling is established in the field of computational fluid dynamics. The origin of modern turbulence theory goes back to Kolmogorov [38, 39]. He introduced the concepts of scale similarity and of a universal inertial cascade. Nevertheless, also before Kolmogorov the description of turbulence was already a subject. It was known for a long time that fluids do not flow smoothly at large scales. This fundamental observation became aware of rivers and clouds. The first quantitative observations of turbulence had been made in the middle of the nineteenth century.

State of the art turbulence models use either algebraic relations or systems of differential or integral equations to describe mathematically the appearance and dissipation of turbulence in high Reynolds number turbulent flows. An important example representing an algebraic turbulence model is the one of Baldwin and Lomax [3]. In this thesis we restrict ourselves to linear turbulence models represented by differential or integral equations. The solutions of these equations are additional quantities in the considered fluid. These occurring variables extend the degrees of freedom given by the conservative variables $\hat{W}$ by a further unknown function

$$\hat{W}_t : \hat{D} \times [0, \hat{T}) \to \mathbb{R}^{N_t}.$$

Here $N_t \in \mathbb{N}$ depends on the turbulence model. In this thesis we have

$$\begin{aligned} N_t &= 1 \qquad \text{for the Spalart-Allmaras model,} \\ N_t &= 2 \qquad \text{for the Wilcox } k\omega\text{-model.} \end{aligned}$$

These additional variables are then used to to determine the eddy viscosity,

$$\mu_t = \mu_t \left(\hat{W}_t \left(\hat{x}, \hat{t} \right), \hat{W} \left(\hat{x}, \hat{t} \right) \right) \geq 0 \quad \text{for all} \quad \left(\hat{x}, \hat{t} \right) \in \hat{D} \times \left[0, \hat{T} \right),$$

required for (2.9). Using (2.14) the additional effective conductivity is computed. The determination of μ_t closes the system of the Reynolds averaged Navier-Stokes equations (2.1a).

There exist plenty of turbulence models. There does not even exist one kind of turbulence model, but throughout the literature there are many kinds of such models. In this thesis we consider the one-equation turbulence model introduced by Spalart and Allmaras [85] and revisited in [1]. As a two-equation model we use the $k\omega$-type model of Wilcox published in 1988 [112]. These two considered models will be presented in the following Sections 2.2.1 and 2.2.2. As preparation we shortly give the following definition.

Definition 2.2.1 *Assume that* $A \in \mathbb{R}^{n \times n}, A = (a_{ij})_{1 \leq i,j \leq n}$ *and* $B \in \mathbb{R}^{n \times n}, B = (b_{ij})_{1 \leq i,j \leq n}$. *Then we define the product*

$$A \otimes B := \sum_{i,j=1}^{n} a_{ij} b_{ij}. \tag{2.17}$$

Lemma 2.2.2 *Assume that* $m = 3$. *Then the following equations hold:*

$$\begin{aligned} \overline{\mathcal{S}} \otimes \frac{d\hat{u}}{d\hat{x}} &= \frac{2}{3} \left\{ \sum_{i=1}^{3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_i} \right)^2 - \sum_{1 \leq i < j \leq 3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_i} \frac{\partial \hat{u}_j}{\partial \hat{x}_j} \right) \right\} + \frac{1}{2} \sum_{1 \leq i < j \leq 3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_j} + \frac{\partial \hat{u}_j}{\partial \hat{x}_i} \right)^2 \\ \Omega \otimes \Omega &= \frac{1}{2} \sum_{1 \leq i < j \leq 3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_j} - \frac{\partial \hat{u}_j}{\partial \hat{x}_i} \right)^2 = \frac{1}{2} \| \mathrm{curl} \left(\hat{u} \right) \|_2^2. \end{aligned}$$

Proof: Both assertions follow by direct computations:

$$\begin{aligned}
\overline{\mathcal{S}} \otimes \frac{d\hat{u}}{d\hat{x}} &= \left\{ \left(\frac{2}{3}\frac{\partial \hat{u}_1}{\partial \hat{x}_1} - \frac{1}{3}\frac{\partial \hat{u}_2}{\partial \hat{x}_2} - \frac{1}{3}\frac{\partial \hat{u}_3}{\partial \hat{x}_3} \right) \frac{\partial \hat{u}_1}{\partial \hat{x}_1} \right. \\
&\quad + \left(-\frac{1}{3}\frac{\partial \hat{u}_1}{\partial \hat{x}_1} + \frac{2}{3}\frac{\partial \hat{u}_2}{\partial \hat{x}_2} - \frac{1}{3}\frac{\partial \hat{u}_3}{\partial \hat{x}_3} \right) \frac{\partial \hat{u}_2}{\partial \hat{x}_2} \\
&\quad \left. + \left(-\frac{1}{3}\frac{\partial \hat{u}_1}{\partial \hat{x}_1} - \frac{1}{3}\frac{\partial \hat{u}_2}{\partial \hat{x}_2} + \frac{2}{3}\frac{\partial \hat{u}_3}{\partial \hat{x}_3} \right) \frac{\partial \hat{u}_3}{\partial \hat{x}_3} \right\} \\
&\quad + \frac{1}{2} \sum_{i,j=1, i\neq j}^{3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_j} + \frac{\partial \hat{u}_j}{\partial \hat{x}_i} \right) \frac{\partial \hat{u}_i}{\partial \hat{x}_j} \\
&= \frac{2}{3} \left\{ \sum_{i=1}^{3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_i} \right)^2 - \left(\frac{\partial \hat{u}_1}{\partial \hat{x}_1}\frac{\partial \hat{u}_2}{\partial \hat{x}_2} + \frac{\partial \hat{u}_1}{\partial \hat{x}_1}\frac{\partial \hat{u}_3}{\partial \hat{x}_3} + \frac{\partial \hat{u}_2}{\partial \hat{x}_2}\frac{\partial \hat{u}_3}{\partial \hat{x}_3} \right) \right\} \\
&\quad + \frac{1}{2} \left\{ \left(\frac{\partial \hat{u}_1}{\partial \hat{x}_2} \right)^2 + \frac{\partial \hat{u}_1}{\partial \hat{x}_2}\frac{\partial \hat{u}_2}{\partial \hat{x}_1} + \left(\frac{\partial \hat{u}_2}{\partial \hat{x}_1} \right)^2 \right. \\
&\quad + \left(\frac{\partial \hat{u}_1}{\partial \hat{x}_3} \right)^2 + \frac{\partial \hat{u}_1}{\partial \hat{x}_3}\frac{\partial \hat{u}_3}{\partial \hat{x}_1} + \left(\frac{\partial \hat{u}_3}{\partial \hat{x}_1} \right)^2 \\
&\quad \left. + \left(\frac{\partial \hat{u}_2}{\partial \hat{x}_3} \right)^2 + \frac{\partial \hat{u}_2}{\partial \hat{x}_3}\frac{\partial \hat{u}_3}{\partial \hat{x}_2} + \left(\frac{\partial \hat{u}_3}{\partial \hat{x}_2} \right)^2 \right\} \\
&= \frac{2}{3} \left\{ \sum_{i=1}^{3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_i} \right)^2 - \sum_{1\leq i<j\leq 3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_i}\frac{\partial \hat{u}_j}{\partial \hat{x}_j} \right) \right\} + \frac{1}{2} \sum_{1\leq i<j\leq 3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_j} + \frac{\partial \hat{u}_j}{\partial \hat{x}_i} \right)^2,
\end{aligned}$$

$$\begin{aligned}
4\,(\Omega \otimes \Omega) &= \sum_{i,j=1}^{3} \left(\frac{\partial \hat{u}_i}{\partial \hat{x}_j} - \frac{\partial \hat{u}_j}{\partial \hat{x}_i} \right)^2 \\
&= 2 \left\{ \left(\frac{\partial \hat{u}_1}{\partial \hat{x}_2} - \frac{\partial \hat{u}_2}{\partial \hat{x}_1} \right)^2 + \left(\frac{\partial \hat{u}_1}{\partial \hat{x}_3} - \frac{\partial \hat{u}_3}{\partial \hat{x}_1} \right)^2 + \left(\frac{\partial \hat{u}_2}{\partial \hat{x}_3} - \frac{\partial \hat{u}_3}{\partial \hat{x}_2} \right)^2 \right\} \\
&= 2\|\mathrm{curl}\,(\hat{u})\,\|_2^2.
\end{aligned}$$

□

2.2.1 Spalart-Allmaras model

The Spalart-Allmaras turbulence model is an established, widely accepted state of the art one-equation turbulence model. Throughout this thesis we follow exactly the recommendations and modifications given in [1, Section 3]. The additional introduced unknown function is

$$\hat{W}_t = \hat{\tilde{\nu}}.$$

Then, for $\hat{t} \in \left[0, \hat{T}\right)$ it needs to satisfy the integral equation

$$V_{\hat{D}}\left(Q_{\mathrm{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right)\right)(\hat{t}) \;=\; \frac{d}{d\hat{t}} V_{\hat{D}}\left(\hat{\tilde{\nu}}\right)(\hat{t}) + R_{\partial\hat{D},\mathrm{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right)(\hat{t}), \qquad (2.18)$$

where the integral operators are

$$\begin{aligned} R_{c,\partial\hat{D},\mathrm{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right)(\hat{t}) &:= \int_{\partial\hat{D}} \left\langle f_{c,\mathrm{SA}}\left(\hat{\tilde{\nu}}\left(\hat{y},\hat{t}\right), \hat{W}\left(\hat{y},\hat{t}\right)\right), n(\hat{y})\right\rangle ds(\hat{y}), \\ R_{v,\partial\hat{D},\mathrm{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right)(\hat{t}) &:= \int_{\partial\hat{D}} \left\langle f_{v,\mathrm{SA}}\left(\hat{\tilde{\nu}}\left(\hat{y},\hat{t}\right) \hat{W}\left(\hat{y},\hat{t}\right)\right), n(\hat{y})\right\rangle ds(\hat{y}), \\ R_{\partial\hat{D},\mathrm{SA}} &:= \hat{R}_{c,\partial\hat{D},\mathrm{SA}} - \hat{R}_{v,\partial\hat{D},\mathrm{SA}}. \end{aligned}$$

The terms $f_{c,\mathrm{SA}}$ and $f_{v,\mathrm{SA}}$ describe the convective and viscous contribution

$$\begin{aligned} f_{c,\mathrm{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right) &:= \hat{\tilde{\nu}}\hat{u}, \\ f_{v,\mathrm{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right) &:= \frac{1}{\sigma}\begin{cases} \left(\nu_l + \hat{\tilde{\nu}}\right) \mathrm{grad}\,\left(\hat{\tilde{\nu}}\right), & \hat{\tilde{\nu}} \geq 0, \\ \left(\nu_l + f_n\hat{\tilde{\nu}}\right) \mathrm{grad}\,\left(\hat{\tilde{\nu}}\right), & \hat{\tilde{\nu}} < 0, \end{cases} \end{aligned}$$

where ν_l denotes the kinematic viscosity (2.13) and the multiplicator f_n is given by

$$\begin{aligned} f_n\left(\hat{\tilde{\nu}}, \hat{W}\right)(\hat{x},\hat{t}) &:= \frac{c_{n1} + \left(\chi\left(\hat{x},\hat{t}\right)\right)^3}{c_{n1} - \left(\chi\left(\hat{x},\hat{t}\right)\right)^3}, \quad c_{n1} := 16, \\ \chi\left(\hat{\tilde{\nu}}, \hat{W}\right) &:= \frac{\hat{\tilde{\nu}}}{\nu_l\left(\hat{W}\right)}. \end{aligned} \qquad (2.19)$$

Given the transported variable $\hat{\tilde{\nu}}$ the eddy viscosity μ_t is computed by

$$\mu_t\left(\hat{\tilde{\nu}}, \hat{W}\right) := \begin{cases} \hat{\rho}\hat{\tilde{\nu}} f_{v_1}, & \hat{\tilde{\nu}} \geq 0, \\ 0, & \hat{\tilde{\nu}} < 0, \end{cases} \qquad f_{v_1}\left(\hat{\tilde{\nu}}, \hat{W}\right) := \frac{\chi^3\left(\hat{\tilde{\nu}}, \hat{W}\right)}{\chi^3\left(\hat{\tilde{\nu}}, \hat{W}\right) + c_{v_1}^3}. \qquad (2.20)$$

To formulate the volume integral operator $V_{\hat{D}}\left(Q_{\mathrm{SA}}\right)$in (2.18) so-called source terms are required. First note that the term Q_{SA} is a sum of three terms,

$$Q_{\mathrm{SA}} \;:=\; Pr_{\mathrm{SA}} - De_{\mathrm{SA}} + Di_{\mathrm{SA}},$$

which are called production, destruction and diffusion source terms. They are given by

$$
\begin{aligned}
Pr_{\mathrm{SA}} &:= \begin{cases} c_{b1}\left(1-f_{t2}\right)\tilde{S}\hat{\tilde{\nu}}, & \hat{\tilde{\nu}} \geq 0 \\ c_{b1}\left(1-c_{t3}\right)S\hat{\tilde{\nu}}, & \hat{\tilde{\nu}} < 0 \end{cases}, \qquad f_{t2} := c_{t3}\exp\left(-c_{t4}\chi^2\right), \\
De_{\mathrm{SA}} &:= \begin{cases} \left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\frac{\hat{\tilde{\nu}}}{\hat{d}}\right)^2, & \hat{\tilde{\nu}} \geq 0 \\ -c_{w1}\left(\frac{\tilde{\nu}}{\hat{d}}\right)^2, & \hat{\tilde{\nu}} < 0 \end{cases}, \quad f_w := g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{1/6} \\
Di_{\mathrm{SA}} &:= \frac{c_{b2}}{\sigma}\left\|\mathrm{grad}\ \hat{\tilde{\nu}}\right\|_2^2, \\
S &:= \sqrt{2\Omega\otimes\Omega}, \quad \overline{S} := \frac{\hat{\tilde{\nu}}}{\kappa^2\hat{d}^2}f_{v2}, \quad \tilde{S} := \begin{cases} S+\overline{S}, & \overline{S} \geq -c_{v2}S \\ S+\frac{S\left(c_{v2}^2S+c_{v3}\overline{S}\right)}{(c_{v3}-2c_{v2})S-\overline{S}}, & \overline{S} < -c_{v2}S \end{cases}, \\
g &:= r + c_{w2}r\left(r^5-1\right), \quad r := \min\left\{\frac{\hat{\tilde{\nu}}}{\kappa^2\hat{d}^2\tilde{S}}, 10\right\}, \quad f_{v2} := 1-\frac{\chi}{1+\chi f_{v1}}, \\
c_{b_1} &:= 0.1355, \quad c_{b_2} := 0.622, \quad \sigma := \frac{2}{3}, \quad \kappa := 0.41, \\
c_{w_1} &:= \frac{c_{b_1}}{\kappa^2} + \frac{1+c_{b_2}}{\sigma}, \quad c_{w_2} := 0.3, \quad c_{w_3} := 2, \\
c_{t_3} &:= 1.2, \quad c_{t_4} := 0.5, \quad c_{v_1} := 7.1, \quad c_{v_2} := 0.7, \quad c_{v_3} := 0.9,
\end{aligned}
$$

and $\hat{d}$ is the distance to the closest wall. Note that due to Lemma 2.2.2 for the three-dimensional case the definition of S can be replaced by

$$S = \|\mathrm{curl}\left(\hat{u}\right)\|_2.$$

2.2.2 $k\omega$-model

Two equations turbulence models are also widely used in the world of computational fluid dynamics. The two differential or integral equations describe quantities for the turbulence kinetic energy and the length scale or dissipation rate. There exist many variants of these models.

A rather complete overview of the number of possible models as well as their relation can be found in the report by Bredberg [8]. In general, they can be classified in the $k\varepsilon$-type and $k\omega$-type models. The dimensional transport variables $\hat{k} = \hat{k}\left(\hat{x},\hat{t}\right)$, $\hat{\varepsilon} = \hat{\varepsilon}\left(\hat{x},\hat{t}\right)$ and $\hat{\omega} = \hat{\omega}\left(\hat{x},\hat{t}\right)$ resp. describe the turbulence kinetic energy and the rate of dissipation of the turbulence kinetic energy. Given the functions $\hat{k}$, $\hat{\varepsilon}$ and $\hat{\omega}$ resp. the required eddy viscosity is then computed either by for example

$$\mu_t = C_\mu\hat{\rho}\frac{\hat{k}^2}{\hat{\varepsilon}}, \qquad C_\mu := \frac{9}{100},$$

or

$$\mu_t = \hat{\rho}\frac{\hat{k}}{\hat{\omega}}. \tag{2.21}$$

Early two equation models rely on the work of Rodi and Spalding [76] and Jones and Launder [32]. Another established two equation model is Menter's Shear-Stress Transport (SST) model [59, 61]. It combines a $k\varepsilon$-type and $k\omega$-type model using an intermediate function such that it behaves in the neighborhood of a no-slip wall like a $k\omega$-type and in the region far away from the wall like a $k\varepsilon$-type model. Also, for the original $k\omega$-type model of Wilcox published in 1988 [112] there has been an update in 2008 [114] where several additional source terms have been introduced.

Besides this general classifications two equation type models possibly differ in their actual formulation. For example:

a) One finds in the literature that the production term in the equation for the turbulence kinetic energy may be formulated using the strain rate or the vorticity.

b) Often one finds that the production term in the equation for the turbulence kinetic energy is limited with respect to the destruction term.

c) Several limitation techniques directly related to the variables k and ω are established.

d) Limitations of the resulting eddy viscosity can be found.

e) Boundary conditions are often not described or different.

f) Parameter choices are different.

With respect to all the variations that can be found, to the author's opinion it can be assumed that there do not exist two computer codes implementing the same two equation turbulence models. And the reason for all these variations is an interesting question on its own.

One answer possibly is that within the context of the compressible RANS equations in combination with a two equation model the robust approximation of a steady state solution is not straightforward. However, being aware of the fact that physical modeling of turbulence is an ongoing task in the field of computational fluid dynamics and that the number of possible two equation models which have been published is large, within this thesis we restrict our investigations to the original $k\omega$-model of Wilcox published in 1988 [112].

Denoting the additional dimensional variables for the $k\omega$-model by

$$\hat{W}_t = \left(\hat{k}, \hat{\omega}\right),$$

the dimensional form of the governing integral equations for the turbulence model is for $\hat{t} \in \left[0, \hat{T}\right)$ expressed by

$$V_{\hat{D}}\left(Q_{(k,\omega)}\left(\hat{W}_t, \hat{W}\right)\right)(\hat{t}) = \frac{d}{d\hat{t}} V_{\hat{D}}\left(\hat{W}_t\right)(\hat{t}) + R_{\partial\hat{D},(k,\omega)}\left(\hat{W}_t, \hat{W}\right)(\hat{t}), \quad (2.22)$$

where the integral operators are

$$\begin{aligned} R_{c,\partial\hat{D},(k,\omega)}\left(\hat{W}_t, \hat{W}\right)(\hat{t}) &:= \int_{\partial\hat{D}} \left\langle f_{c,(k,\omega)}\left(\hat{W}_t(\hat{y},\hat{t}), \hat{W}(\hat{y},\hat{t})\right), n(\hat{y})\right\rangle ds(\hat{y}), \\ R_{v,\partial\hat{D},(k,\omega)}\left(\hat{W}_t, \hat{W}\right)(\hat{t}) &:= \int_{\partial\hat{D}} \left\langle f_{v,(k,\omega)}\left(\hat{W}_t(\hat{y},\hat{t}), \hat{W}(\hat{y},\hat{t})\right), n(\hat{y})\right\rangle ds(\hat{y}), \\ R_{\partial\hat{D},(k,\omega)} &:= \hat{R}_{c,\partial\hat{D},(k,\omega)} - \hat{R}_{v,\partial\hat{D},(k,\omega)}. \end{aligned}$$

Here the convective $f_{c,(k,\omega)}$ and viscous $f_{v,(k,\omega)}$ contributions as well as the source terms $Q_{(k,\omega)}$ are given by

$$\begin{aligned} f_{c,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) &:= \begin{pmatrix} \hat{k}\hat{u} \\ \hat{\omega}\hat{u} \end{pmatrix}, \\ f_{v,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) &:= \begin{pmatrix} \left(\nu_l + \sigma_k \frac{\hat{k}}{\hat{\omega}}\right) \text{grad } \hat{k} \\ \left(\nu_l + \sigma_\omega \frac{\hat{k}}{\hat{\omega}}\right) \text{grad } \hat{w} \end{pmatrix}, \\ Q_{(k,\omega)}\left(\hat{W}_t, \hat{W}\right) &:= \begin{pmatrix} Pr_{k,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) - De_{k,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) \\ Pr_{\omega,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) - De_{\omega,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) \end{pmatrix}. \end{aligned}$$

The kinematic viscosity ν_l is defined in (2.13). The production and destruction terms required to formulate the source terms are

$$\begin{aligned} Pr_{k,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) &:= 2\frac{\hat{k}}{\hat{\omega}} \overline{\mathcal{S}}(\hat{u}) \otimes \frac{d\hat{u}}{d\hat{x}}, \qquad De_{k,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) := \beta^* \hat{k}\hat{\omega} \\ Pr_{\omega,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) &:= \alpha \overline{\mathcal{S}}(\hat{u}) \otimes \frac{d\hat{u}}{d\hat{x}}, \qquad De_{\omega,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) := \beta \hat{\omega}^2. \end{aligned}$$

The term $\overline{\mathcal{S}} \otimes \frac{du}{dx}$ is explicitly given in Lemma 2.2.2. Often, in the literature one also finds

$$Pr_{\omega,(k,\omega)} = \alpha \frac{\hat{\omega}}{\hat{k}} Pr_{k,(k,\omega)} = \alpha \nu_t^{-1} Pr_{k,(k,\omega)} \quad (2.23)$$

as definition, where ν_t is given by (2.16). Though such definition is analytically equivalent, within an implementation care must be taken, since k tends to have very small values near zero. This may cause loss of stability of the algorithm. Furthermore, we also do not use the clean production term stated above, but the analytic representation is replaced by

$$\tilde{Pr}_{k,(k,\omega)} := \min\left\{Pr_{k,(k,\omega)}, 20 \cdot De_{k,(k,\omega)}\right\}. \quad (2.24)$$

We will show in example in Section 6.11.1 the necessity for (2.24).

2.3 Nondimensionalization

The choice of nondimensional parameters is arbitrary. Here we follow the Buckingham π theorem.

Theorem 2.3.1 *Assume that $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are physical quantities measured in a consistent system of units, that is the SI system (International System of Units). The basic units are the meter, kilogram, second, ampere, and kelvin* (m, kg, sec, A, K). *Then any physically meaningful relation*

$$\Phi(\mathcal{P}_1, \ldots, \mathcal{P}_n) = 0, \qquad \mathcal{P}_j \neq 0, \quad j = 1, \ldots, n,$$

is equivalent to a relation of the form

$$\Psi(\pi_1, \ldots, \pi_{n-r}) = 0$$

involving a maximal set of independent dimensionless combinations.

Proof: See for example the textbook of Bluman and Kumei [7].

□

Roughly speaking, Theorem 2.3.1 states that if the equation of a physical law involves n physical variables, and the rank of the matrix of dimensions is k, then the original equation is equivalent to an equation involving a set of $n-k$ dimensionless parameters constructed from the original variables.

To apply Theorem 2.3.1 to the equation (2.1a) we choose the five following independent variables density $\hat{\rho}$, pressure $\hat{p}$, temperature $\hat{T}$, velocity $\hat{u}$, and the universal gas constant $\hat{\Re}$. The dimension matrix of the these quantities is given in Table 2.1. To

	$\hat{\rho}$	$\hat{p}$	$\hat{T}$	$\hat{u}$	$\hat{\Re}$
kg	1	1	0	0	0
m	-3	-1	0	1	2
sec	0	-2	0	-1	-2
K	0	0	1	0	1

Table 2.1: Dimensional matrix

find a dimensionless combination of $\hat{\rho}, \hat{p}, \hat{T}, \hat{u}$, and $\hat{\Re}$ we need to determine $\lambda_1, \ldots \lambda_5$ such that

$$1 = \left[\hat{\rho}^{\lambda_1}\hat{p}^{\lambda_2}\hat{T}^{\lambda_3}\hat{u}^{\lambda_4}\hat{\Re}^{\lambda_5}\right] = \left(\frac{\mathrm{kg}}{\mathrm{m}^3}\right)^{\lambda_1}\left(\frac{\mathrm{kg}}{\mathrm{m}\cdot\mathrm{sec}^2}\right)^{\lambda_2}\mathrm{K}^{\lambda_3}\left(\frac{\mathrm{m}}{\mathrm{sec}}\right)^{\lambda_4}\left(\frac{\mathrm{m}^2}{\mathrm{sec}^2\cdot\mathrm{K}}\right)^{\lambda_5},$$

which is equivalent to find nontrivial solutions of the linear system

$$A_{\text{phys}}\lambda = 0, \qquad \text{where} \qquad A_{\text{phys}} := \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -3 & -1 & 0 & 1 & 2 \\ 0 & -2 & 0 & -1 & -2 \\ 0 & 0 & 1 & 0 & -1 \end{pmatrix}. \tag{2.25}$$

Theorem 2.3.2 *The matrix A_{phys} has rank 3 and a basis of the null space $\mathcal{N}(A_{\text{phys}})$ is given by*

$$v_1 := \begin{pmatrix} -1 \\ 1 \\ -1 \\ 0 \\ -1 \end{pmatrix} \qquad \textit{and} \qquad v_2 := \begin{pmatrix} 0 \\ 0 \\ 1 \\ -2 \\ 1 \end{pmatrix}.$$

Proof: The theorem follows by straightforward computations.

□

So, using Theorem 2.3.2 according to the chosen basis vectors of $\mathcal{N}(A_{\text{phys}})$ dimensionless combinations are

$$1 = \frac{[\hat{p}]}{[\hat{\rho}]\left[\hat{T}\right]\left[\hat{\Re}\right]} \tag{2.26a}$$

$$1 = \frac{\left[\hat{T}\right]\left[\hat{\Re}\right]}{[\hat{u}]^2}. \tag{2.26b}$$

And, due to Theorem 2.3.2 the combinations (2.26) represent a maximal set of independent dimensionless combinations. To nondimensionalize the variables, assume that reference states with dimensions $\rho_{\text{ref}}, p_{\text{ref}}, T_{\text{ref}}$ are given. Then, using (2.26a) and (2.26b) we obtain

$$\Re_{\text{ref}} = \frac{p_{\text{ref}}}{\rho_{\text{ref}} T_{\text{ref}}} \qquad \text{and} \qquad u_{\text{ref}} = \sqrt{T_{\text{ref}}\Re_{\text{ref}}} = \sqrt{\frac{p_{\text{ref}}}{\rho_{\text{ref}}}}. \tag{2.27}$$

Hence, non-dimensionalized variables, free stream values, and additionally the length scale and time may be obtained by

$$\begin{aligned} \rho &:= \hat{\rho}/\rho_{\text{ref}}, & \rho_\infty &:= \hat{\rho}_\infty/\rho_{\text{ref}}, \\ p &:= \hat{p}/p_{\text{ref}}, & p_\infty &:= \hat{p}_\infty/p_{\text{ref}}, \\ T &:= \hat{T}/T_{\text{ref}}, & T_\infty &:= \hat{T}_\infty/T_{\text{ref}}, \\ L &:= \hat{L}/L_{\text{ref}}, & t &:= L/u. \end{aligned} \tag{2.28a}$$

For simplicity we choose $\rho_{\text{ref}} = \hat{\rho}_\infty$, $p_{\text{ref}} = \hat{p}_\infty$, and $T_{\text{ref}} = \hat{T}_\infty$. Due to this choice the reference kinematic and laminar viscosity are given by

$$\nu_{l,\text{ref}} = u_{\text{ref}} L_{\text{ref}}, \qquad \mu_{l,\text{ref}} = \rho_{\text{ref}} \nu_{l,\text{ref}}. \tag{2.29}$$

As a consequence, we obtain the following normalized nondimensionalized relations

$$\begin{aligned}
\rho_\infty &= 1, && (2.30a)\\
p_\infty &= 1,\\
T_\infty &= 1,\\
\Re &= 1,\\
u_\infty &= \frac{\hat{u}_\infty}{\sqrt{\frac{\hat{p}_{\text{ref}}}{\hat{\rho}_{\text{ref}}}}} = \frac{M_\infty \hat{a}_\infty}{\sqrt{\frac{\hat{p}_{\text{ref}}}{\hat{\rho}_{\text{ref}}}}} = \frac{M_\infty \sqrt{\gamma \frac{\hat{p}_\infty}{\hat{\rho}_\infty}}}{\sqrt{\frac{\hat{p}_{\text{ref}}}{\hat{\rho}_{\text{ref}}}}} = M_\infty \sqrt{\gamma}, && (2.30b)\\
a_\infty &= \sqrt{\frac{\gamma p_\infty}{\rho_\infty}} = \sqrt{\gamma},\\
\mu_{l,\infty} &= \frac{\rho_\infty u_\infty L}{Re} = \frac{\sqrt{\gamma} M_\infty L}{Re}. && (2.30c)
\end{aligned}$$

To this end Sutherland's law (2.10) in its nondimensionalized version is implemented by

$$\begin{aligned}
\mu_l &= \frac{\sqrt{\gamma} M_\infty L}{Re} \left(\frac{\hat{T}}{T_{\text{ref}}} \right)^{3/2} \left(\frac{\frac{\hat{T}_\infty}{T_{\text{ref}}} + \frac{\hat{\tilde{T}}}{T_{\text{ref}}}}{\frac{\hat{T}}{T_{\text{ref}}} + \frac{\hat{\tilde{T}}}{T_{\text{ref}}}} \right) = \frac{\sqrt{\gamma} M_\infty L}{Re} \Gamma(T), && (2.31a)\\
\Gamma(T) &:= T^{3/2} \left(\frac{1 + C_{\text{suth}}}{T + C_{\text{suth}}} \right), \qquad C_{\text{suth}} := \frac{\hat{\tilde{T}}}{T_{\text{ref}}}. && (2.31b)
\end{aligned}$$

The implementation of (2.31b) requires both the temperature $\hat{\tilde{T}}$ defined in (2.12) and the reference temperature T_{ref} or the relation C_{suth}. Both must be given for a viscous computation. In general we choose $T_{\text{ref}} = 273.15\text{K}$. Furthermore, note that the velocity u represents a three dimensional vector field

$$\begin{aligned}
u : D \times [0, T) &\to \mathbb{R}^m\\
(x, t) &\mapsto \begin{pmatrix} u_1(x,t) \\ \vdots \\ u_m(x,t) \end{pmatrix}
\end{aligned}$$

and the equation (2.30b) is understood that $u_\infty = \|u_\infty\|_2 = M_\infty \sqrt{\gamma}$.

Not only the variables need to be nondimensionalized, in general the domain $\hat{D}$ describes the physical domain and needs to be scaled to the nondimensional domain. Hence, whole integral equation (2.1a) has to be reformulated in nondimensional form. In a first step the differential operators and integrals need to be represented with respect to the nondimensional length scale L.

Definition 2.3.3 *Assume that $D \subset \mathbb{R}^m$. The mapping*

$$\begin{aligned} g_1 : D &\rightarrow \hat{D}, \\ x &\mapsto \hat{x} = L_{\text{ref}} x, \end{aligned}$$

describes in the following the conversion of the scaled domain D by the reference length L_{ref} to the original domain $\hat{D}$. The mapping

$$\begin{aligned} g_2 : [0, T) &\rightarrow [0, \hat{T}), \\ t &\mapsto \hat{t} = \frac{L_{\text{ref}}}{u_{\text{ref}}} t, \end{aligned}$$

describes the conversion of time to nondimensional time. Finally we define the mapping $g(x, t) = (g_1(x), g_2(t))$.

Lemma 2.3.4 *The differential operators satisfy*

$$\frac{d}{d\hat{t}} = \frac{u_{\text{ref}}}{L_{\text{ref}}} \frac{d}{dt}, \tag{2.32}$$

$$\frac{\partial}{\partial \hat{x}_j} = \frac{1}{L_{\text{ref}}} \frac{\partial}{\partial x_j}. \tag{2.33}$$

Proof: We define the mapping $h(t) = t_{\text{ref}} t = \frac{L_{\text{ref}}}{u_{\text{ref}}} t$ (see (2.30c)). Hence, we have

$$\frac{dh(t)}{dt} = \frac{L_{\text{ref}}}{u_{\text{ref}}}, \qquad \frac{dh^{-1}(\hat{t})}{d\hat{t}} = \frac{u_{\text{ref}}}{L_{\text{ref}}},$$

which proves (2.32). The partial derivative of g_1 of Definition 2.3.3 is given by

$$\frac{\partial g(x)}{\partial x_j} = L_{\text{ref}}, \qquad \frac{\partial g^{-1}(x)}{\partial x_j} = \frac{1}{L_{\text{ref}}},$$

which proves (2.33).

□

Lemma 2.3.5 *Let g_1 be given by Definition 2.3.3 and let us assume that (2.28a) holds. Then we have*

$$\int_{\hat{D}} v(x) dx = L_{\text{ref}}^m \int_D v\left(g_1(y)\right) dy, \tag{2.34}$$

$$\int_{\partial \hat{D}} \langle v(y), n(y) \rangle \, ds(y) = L_{\text{ref}}^{m-1} \int_{\partial D} \langle v(g_1(y)), n(y) \rangle \, ds(y). \tag{2.35}$$

Proof: Equation (2.34) follows by the integral substitution formula

$$\int_{\hat{D}} v(x)dx = \int_{g^{-1}\left(\hat{D}\right)} v\left(g(y)\right) \left|\det\left(\frac{dg(y)}{dy}\right)\right| dy = L_{\text{ref}}^m \int_D v\left(g(y)\right) dy.$$

The second assertion (2.35) is a consequence of Gauss' integral theorem, equation (2.35) and (2.33):

$$\begin{aligned}
\int_{\partial\hat{D}} \langle v(y), n(y)\rangle \, ds(y) &= \int_{\hat{D}} \operatorname{div}_x \left(v(x)\right) dx = L_{\text{ref}}^m \int_D \operatorname{div}_g \left(v\left(g(y)\right)\right) dy \\
&= L_{\text{ref}}^m \int_D \sum_{j=1}^m \frac{\partial v\left(g(y)\right)}{\partial g_j} \frac{\partial g(y)}{\partial y_j} \left(\frac{\partial g(y)}{\partial y_j}\right)^{-1} dy \\
&= L_{\text{ref}}^{m-1} \int_D \operatorname{div}_y v\left(g(y)\right) dy \\
&= L_{\text{ref}}^{m-1} \int_{\partial D} \langle v(g(y)), n(y)\rangle \, ds(y).
\end{aligned}$$

□

To formulate the integral equation (2.1a) and the necessary operators (2.1b)– (2.1d) with respect to nondimensional quantities, we define the operator converting the dimensional variables $\hat{W}$ into nondimensional variables W,

$$\begin{aligned}
N_W &:= \operatorname{diag}\left(\rho_{\text{ref}}, \rho_{\text{ref}} u_{\text{ref}}, \ldots, \rho_{\text{ref}} u_{\text{ref}}, \rho_{\text{ref}} u_{\text{ref}}^2\right), \\
W &:= N_W^{-1} \hat{W}.
\end{aligned}$$

Then we obtain for the volume operator $V_{\hat{D}}$ the relation

$$V_{\hat{D}}\left(\hat{W}\right)\left(\hat{t}\right) = \int_{\hat{D}} \hat{W}\left(\hat{x}, \hat{t}\right) d\hat{x} = N_W \int_{\hat{D}} N_W^{-1} \hat{W}\left(\hat{x}, \hat{t}\right) d\hat{x} = N_W V_{\hat{D}}\left(W\right)\left(\hat{t}\right). \quad (2.36)$$

For the integral operator for the convective fluxes we have

$$\begin{aligned}
R_{c,\partial\hat{D}}\left(\hat{W}\right)\left(\hat{t}\right) &= \int_{\partial\hat{D}} \left\langle f_c\left(N_W N_W^{-1} \hat{W}\left(\hat{y}, \hat{t}\right)\right), n\left(\hat{y}\right)\right\rangle ds\left(\hat{y}\right) \\
&= u_{\text{ref}} N_W \int_{\partial\hat{D}} \left\langle f_c\left(W\left(\hat{y}, \hat{t}\right)\right), n\left(\hat{y}\right)\right\rangle ds(\hat{y}) \\
&= u_{\text{ref}} N_W R_{c,\partial\hat{D}}\left(W\right)\left(\hat{t}\right). \qquad (2.37)
\end{aligned}$$

To formulate the viscous fluxes in nondimensional form we use

$$\begin{aligned}
\overline{\mathcal{S}}\left(\hat{u}\right) &= \overline{\mathcal{S}}\left(u_{\text{ref}} u_{\text{ref}}^{-1} \hat{u}\right) = u_{\text{ref}} \overline{\mathcal{S}}\left(u\right), \\
\mu_{l,\infty}\left(\hat{W}\right) &= \rho_{\text{ref}} u_{\text{ref}} L_{\text{ref}} \frac{\hat{\rho}_\infty \hat{u}_\infty \hat{L}}{\rho_{\text{ref}} u_{\text{ref}} L_{\text{ref}}} \frac{1}{Re} = \rho_{\text{ref}} u_{\text{ref}} L_{\text{ref}} \frac{\sqrt{\gamma} M_\infty L}{Re}. \qquad (2.38)
\end{aligned}$$

The second equation is a consequence of (2.10), (2.30a) and (2.30b). Then, for the viscous stress tensor we have

$$\begin{aligned}\tau\left(\hat{W}\right) &= 2\rho_{\mathrm{ref}}u_{\mathrm{ref}}^2L_{\mathrm{ref}}\frac{\sqrt{\gamma}M_\infty L}{Re}\Gamma(T)\overline{\mathcal{S}}\left(u\right)=2\rho_{\mathrm{ref}}u_{\mathrm{ref}}^2L_{\mathrm{ref}}\mu_l\left(W\right)\overline{\mathcal{S}}\left(u\right)\\ &= \rho_{\mathrm{ref}}u_{\mathrm{ref}}^2L_{\mathrm{ref}}\tau\left(W\right).\end{aligned}\tag{2.39}$$

Furthermore, using (2.27), (2.31) and (2.38) the laminar conductivity reads

$$\begin{aligned}\kappa_l\left(\hat{W}\right) &= \frac{\Re_{\mathrm{ref}}}{\Re_{\mathrm{ref}}}\hat{\Re}\frac{\gamma}{\gamma-1}\rho_{\mathrm{ref}}u_{\mathrm{ref}}L_{\mathrm{ref}}\frac{\sqrt{\gamma}M_\infty L}{Re}\frac{\Gamma(T)}{Pr_l}\\ &= \frac{\rho_{\mathrm{ref}}u_{\mathrm{ref}}^3L_{\mathrm{ref}}}{T_{\mathrm{ref}}}\frac{c_p}{Pr_l}\frac{\sqrt{\gamma}M_\infty L}{Re}\Gamma(T)\\ &= \frac{\rho_{\mathrm{ref}}u_{\mathrm{ref}}^3L_{\mathrm{ref}}}{T_{\mathrm{ref}}}\kappa_l\left(W\right).\end{aligned}\tag{2.40}$$

Hence, we obtain

$$q\left(\hat{W}\right)=\rho_{\mathrm{ref}}u_{\mathrm{ref}}^3L_{\mathrm{ref}}\kappa_l\left(W\right)\mathrm{grad}\;T=\rho_{\mathrm{ref}}u_{\mathrm{ref}}^3L_{\mathrm{ref}}q\left(W\right).\tag{2.41}$$

Equation (2.39) together with (2.41) and (2.8) yields

$$\theta\left(\hat{W}\right) = \rho_{\mathrm{ref}}u_{\mathrm{ref}}^3L_{\mathrm{ref}}\left[\tau\left(W\right)u+q\right]=\rho_{\mathrm{ref}}u_{\mathrm{ref}}^3L_{\mathrm{ref}}\theta\left(W\right).\tag{2.42}$$

An application of (2.39) and (2.42) gives the equation

$$R_{v,\partial\hat{D}}\left(\hat{W}\right)\left(\hat{t}\right)=L_{\mathrm{ref}}u_{\mathrm{ref}}N_WR_{v,\partial\hat{D}}\left(W\right)\left(\hat{t}\right).\tag{2.43}$$

Note that so far the transformations (2.36), (2.37), (2.39), (2.42) and (2.43) do not include the conversion of the integral and differential operators. A straightforward consideration using Definition 2.3.3, (2.36), (2.32) and (2.34) gives

$$\frac{d}{d\hat{t}}V_{\hat{D}}\left(\hat{W}\right)\left(\hat{t}\right)=u_{\mathrm{ref}}L_{\mathrm{ref}}^{m-1}N_W\frac{d}{dt}V_D\left(W\circ g\right)\left(t\right).\tag{2.44a}$$

Similar, using (2.35) and (2.37) we obtain

$$R_{c,\partial\hat{D}}\left(\hat{W}\right)\left(\hat{t}\right)=u_{\mathrm{ref}}L_{\mathrm{ref}}^{m-1}N_WR_{c,\partial D}\left(W\circ g\right)\left(t\right).\tag{2.44b}$$

To reformulate (2.43) we use (2.33) and (2.35) such that

$$R_{v,\partial\hat{D}}\left(\hat{W}\right)\left(\hat{t}\right)=u_{\mathrm{ref}}L_{\mathrm{ref}}^{m-1}N_WR_{v,\partial D}\left(W\circ g\right)\left(t\right).\tag{2.44c}$$

The equations (2.1a) and (2.45) only differ in the factor $u_{\text{ref}} L_{\text{ref}}^{m-1} N_W$. Concluding, the representations (2.44a)– (2.44c) show that the original system of equations defined by (2.1) can be equivalently reformulated by

$$0 \;=\; \frac{d}{dt} V_D \left(W \circ g\right)(t) + R_{\partial D}\left(W \circ g\right)(t), \qquad t \in \left[0, g_2\left(T\right)\right). \tag{2.45}$$

As next step we need to consider the nondimensionalization of the additional system of equations defining the eddy viscosity μ_t and eddy conductivity κ_t in (2.9). In principle the system of equations (2.18) and (2.22) can be nondimensionalized independently of (2.1). Hence, at the outset we are free to choose $\tilde{\nu}_{\text{ref}}$, k_{ref} and ω_{ref}. On the other hand there exist conditions which need to be satisfied. For example, naturally the computed eddy viscosity μ_t needs to be compatible with (2.9), that is with (2.31a).

We start with the Spalart-Allmaras turbulence model (2.18). Using the compatability (2.9) and comparing (2.20) and (2.31a) we conclude

$$\left[\hat{\rho}\hat{u}\hat{L}\right] = \left[\mu_l\right] = \left[\mu_t\right] = \left[\hat{\rho}\hat{\tilde{\nu}}\hat{f}_{v1}\right], \quad \text{hence} \quad \left[\hat{u}\hat{L}\right] = \left[\hat{\tilde{\nu}} f_{v1}\right].$$

This simple consideration suggests to choose $\tilde{\nu}_{\text{ref}}$ such that f_{v1} is dimensionless, that is $\left[f_{v1}\right] = 1$. Using the definition of f_{v1} and χ in (2.19) this can be reached by

$$\tilde{\nu}_{\text{ref}} := u_{\text{ref}} L_{\text{ref}}. \tag{2.46}$$

The choice (2.46) corresponds directly to (2.29). Hence, the nondimensional $\tilde{\nu}$ for the Spalart-Allmaras model is chosen to be

$$\tilde{\nu} = \frac{\hat{\tilde{\nu}}}{\tilde{\nu}_{\text{ref}}} = \frac{\hat{\tilde{\nu}}}{u_{\text{ref}} L_{\text{ref}}}.$$

Then, following the modifications in (2.36) and (2.44a) we conclude

$$\frac{d}{d\hat{t}} V_{\hat{D}}\left(\hat{\tilde{\nu}}\right)\left(\hat{t}\right) = u_{\text{ref}} L_{\text{ref}} \frac{d}{d\hat{t}} \int_{\hat{D}} \tilde{\nu}\left(\hat{x}, \hat{t}\right) d\hat{x} = u_{\text{ref}}^2 L_{\text{ref}}^m \frac{d}{dt} V_D\left(\tilde{\nu} \circ g\right)(t). \tag{2.47}$$

And, similar straightforward considerations give using (2.33) additionally for the viscous term

$$R_{c,\partial\hat{D},\text{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right)\left(\hat{t}\right) \;=\; u_{\text{ref}}^2 L_{\text{ref}}^m R_{c,\partial D,\text{SA}}\left(\tilde{\nu} \circ g, W \circ g\right)(t), \tag{2.48}$$

$$R_{v,\partial\hat{D},\text{SA}}\left(\hat{\tilde{\nu}}, \hat{W}\right)\left(\hat{t}\right) \;=\; u_{\text{ref}}^2 L_{\text{ref}}^m R_{v,\partial D,\text{SA}}\left(\tilde{\nu} \circ g, W \circ g\right)(t). \tag{2.49}$$

To obtain the shape of the source terms with respect to nondimensional variables, first note that the constants of the model, for example $c_{b1}, c_{t3}, \ldots$ as well as f_{v1}, f_{v2}

and f_{t2} are nondimensional quantities since χ is nondimensional. The nondimensional distance to the closest wall is given by $d = \hat{d}/L_{\text{ref}}$. Then, straightforward computation show

$$\begin{aligned}
S(\hat{u}) &= \frac{u_{\text{ref}}}{L_{\text{ref}}} S(u), \qquad \overline{S}(\hat{u}) = \frac{u_{\text{ref}}}{L_{\text{ref}}} \overline{S}(u), \\
\tilde{S}(\hat{u}) &= \begin{cases} \frac{u_{\text{ref}}}{L_{\text{ref}}} \left(S(u) + \overline{S}(u) \right), & \frac{u_{\text{ref}}}{L_{\text{ref}}} \overline{S}(u) \geq -c_{v2} \frac{u_{\text{ref}}}{L_{\text{ref}}} S(u) \\ \frac{u_{\text{ref}}}{L_{\text{ref}}} S(u) + \frac{\frac{u_{\text{ref}}}{L_{\text{ref}}} S(u) \left(c_{v2}^2 \frac{u_{\text{ref}}}{L_{\text{ref}}} S(u) + c_{v3} \frac{u_{\text{ref}}}{L_{\text{ref}}} \overline{S}(u) \right)}{\frac{u_{\text{ref}}}{L_{\text{ref}}} \left((c_{v3} - 2c_{v2}) S(u) - \overline{S}(u) \right)}, & \frac{u_{\text{ref}}}{L_{\text{ref}}} \overline{S}(u) < -c_{v2} \frac{u_{\text{ref}}}{L_{\text{ref}}} S(u) \end{cases} \\
&= \frac{u_{\text{ref}}}{L_{\text{ref}}} \begin{cases} \left(S(u) + \overline{S}(u) \right), & \overline{S}(u) \geq -c_{v2} S(u) \\ \left(S(u) + \frac{S(u) \left(c_{v2}^2 S(u) + c_{v3} \overline{S}(u) \right)}{(c_{v3} - 2c_{v2}) S(u) - \overline{S}(u)} \right), & \overline{S}(u) < -c_{v2} S(u) \end{cases} \\
&= \frac{u_{\text{ref}}}{L_{\text{ref}}} \tilde{S}(u). \qquad (2.50)
\end{aligned}$$

Inserting (2.50) into the production term we get

$$Pr_{\text{SA}} \left(\hat{\tilde{\nu}}, \hat{W} \right) = u_{\text{ref}}^2 Pr_{\text{SA}} \left(\tilde{\nu}, W \right).$$

Similar, for the diffusion and destruction term we have

$$\begin{aligned}
Di_{\text{SA}} \left(\hat{\tilde{\nu}}, \hat{W} \right) &= u_{\text{ref}}^2 \left\| \text{grad } \tilde{\nu} \right\|^2 = u_{\text{ref}}^2 Di_{\text{SA}} \left(\tilde{\nu}, W \right) \\
De \left(\hat{\tilde{\nu}}, \hat{W} \right) &= u_{\text{ref}}^2 De \left(\tilde{\nu}, W \right),
\end{aligned}$$

where we have exploited that r and g are also nondimensional quantities by construction,

$$r \left(\hat{\tilde{\nu}}, \hat{W} \right) = \min \left\{ \frac{u_{\text{ref}} L_{\text{ref}} \tilde{\nu}}{\kappa^2 L_{ref}^2 d^2 \frac{u_{\text{ref}}}{L_{\text{ref}}} \tilde{S}(u)}, 10 \right\} = \min \left\{ \frac{\tilde{\nu}}{\kappa^2 d^2 \tilde{S}(u)}, 10 \right\} = r \left(\tilde{\nu}, W \right).$$

In summary, the volume operator for the source terms is reformulated as

$$V_{\hat{D}} \left(Q_{\text{SA}} \left(\hat{\tilde{\nu}}, \hat{W} \right) \right) \left(\hat{t} \right) = u_{\text{ref}}^2 L_{\text{ref}}^m V_D \left(Q_{\text{SA}} \left(\tilde{\nu} \circ g, W \circ g \right) \right) (t). \qquad (2.51)$$

So, using (2.47), (2.48), (2.49) and (2.51) the dimensional equation (2.18) is equivalent to the equation for the nondimensional quantity $\tilde{\nu}$,

$$\begin{aligned}
& V_D \left(Q_{\text{SA}} \left(\tilde{\nu} \circ g, W \circ g \right) \right) (t) \\
= \; & \frac{d}{dt} V_D \left(\tilde{\nu} \circ g \right) (t) + R_{\partial D, \text{SA}} \left(\tilde{\nu} \circ g, W \circ g \right) (t), \quad t \in [0, T). \qquad (2.52)
\end{aligned}$$

Finally, considerations for the nondimensionalization of the $k\omega$-model are required. We start with the simple consideration just as in case of the Spalart-Allmaras model. Using the compatability (2.9) and comparing (2.21) and (2.31a) we conclude

$$\left[\hat{\rho} \hat{u} \hat{L} \right] = [\mu_l] = [\mu_t] = \left[\hat{\rho} \hat{k} \hat{\omega}^{-1} \right], \quad \text{hence} \quad \left[\hat{u} \hat{L} \right] = \left[\hat{k} \hat{\omega}^{-1} \right].$$

With respect to the knowledge that $\hat{k}$ represents the turbulence kinetic energy and $\hat{\omega}$ a length scale, the dimensions of these variables are

$$\left[\hat{k}\right] = \frac{\mathrm{m}^2}{\mathrm{sec}^2} \qquad \text{and hence} \qquad [\hat{\omega}] = \frac{1}{\mathrm{sec}}.$$

So, to nondimensionalize $\hat{k}$ and $\hat{\omega}$ we choose as reference values

$$k_{\mathrm{ref}} = u_{\mathrm{ref}}^2 \qquad \text{and} \qquad \omega_{\mathrm{ref}} = \frac{u_{\mathrm{ref}}}{L_{\mathrm{ref}}}.$$

Then nondimensional variables, using an additional scaling for ω, may be given by

$$k = \frac{\hat{k}}{u_{\mathrm{ref}}^2} \qquad \text{and} \qquad \omega = \frac{\hat{\omega}}{\omega_{sc}\omega_{\mathrm{ref}}}, \qquad \omega_{sc} = \frac{Re}{\sqrt{\gamma}M_\infty L}.$$

The choice of ω_{sc} is explained later. Then, using

$$N_{W_t,(k,\omega)} = \begin{pmatrix} k_{\mathrm{ref}} & 0 \\ 0 & \omega_{\mathrm{ref}} \end{pmatrix}$$

a straightforward computation gives

$$\frac{d}{d\hat{t}} V_{\hat{D}}\left(\hat{W}_t\right)(\hat{t}) = u_{\mathrm{ref}} L_{\mathrm{ref}}^{m-1} \begin{pmatrix} 1 & 0 \\ 0 & \omega_{sc} \end{pmatrix} N_{W_t,(k,\omega)} \frac{d}{dt} V_D\left(W_t \circ g\right)(t),$$

and

$$\begin{aligned} & R_{c,\partial\hat{D},(k,\omega)}\left(\hat{W}_t, \hat{W}\right)(\hat{t}) \\ = \; & u_{\mathrm{ref}} L_{\mathrm{ref}}^{m-1} \begin{pmatrix} 1 & 0 \\ 0 & \omega_{sc} \end{pmatrix} N_{W_t,(k,\omega)} R_{c,\partial D,(k,\omega)}\left(W_t \circ g, W \circ g\right)(t). \end{aligned}$$

Next we consider the viscous contribution for the $k\omega$-model. In a first step we rewrite each of the occurring terms,

$$\begin{aligned} \nu_l\left(\hat{W}\right) \mathrm{grad}\, \hat{k} &= u_{\mathrm{ref}} L_{\mathrm{ref}} \frac{\sqrt{\gamma}M_\infty L \Gamma(T)}{Re} u_{\mathrm{ref}}^2 \mathrm{grad}\, k = u_{\mathrm{ref}}^3 L_{\mathrm{ref}} w_{sc}^{-1} \Gamma(T) \mathrm{grad}\, k, \\ \nu_l\left(\hat{W}\right) \mathrm{grad}\, \hat{\omega} &= u_{\mathrm{ref}} L_{\mathrm{ref}} \frac{\sqrt{\gamma}M_\infty L \Gamma(T)}{Re} \omega_{sc} \frac{u_{\mathrm{ref}}}{L_{\mathrm{ref}}} \mathrm{grad}\, \omega = u_{\mathrm{ref}}^2 \Gamma(T) \mathrm{grad}\, \omega \\ \sigma_k \frac{\hat{k}}{\hat{\omega}} \mathrm{grad}\, \hat{k} &= \sigma_k u_{\mathrm{ref}}^3 L_{\mathrm{ref}} \left(\omega_{sc}\right)^{-1} \frac{k}{\omega} \mathrm{grad}\, k \\ \sigma_\omega \frac{\hat{k}}{\hat{\omega}} \mathrm{grad}\, \hat{\omega} &= \sigma_\omega u_{\mathrm{ref}}^2 \frac{k}{\omega} \mathrm{grad}\, \omega, \end{aligned}$$

and hence

$$f_{v,(k,\omega)}\left(\hat{W}_t, \hat{W}\right) = u_{\mathrm{ref}} L_{\mathrm{ref}} N_{W_t,(k,\omega)} \begin{pmatrix} w_{sc}^{-1} & 0 \\ 0 & 1 \end{pmatrix} \tilde{f}_{v,(k,\omega)}$$

where

$$\tilde{f}_{v,(k,\omega)}\left(W_t, W\right) := \begin{pmatrix} \left(\Gamma(T) + \sigma_k \frac{k}{\omega}\right) \text{grad } k \\ \left(\Gamma(T) + \sigma_\omega \frac{k}{\omega}\right) \text{grad } w \end{pmatrix}.$$

For the corresponding boundary integral operator we get

$$\begin{aligned} & \tilde{R}_{v,\partial\hat{D},(k,\omega)}\left(\hat{W}_t, \hat{W}\right)(\hat{t}) \\ = \; & u_{\text{ref}} L_{\text{ref}}^{m-1} N_{W_t,(k,\omega)} \begin{pmatrix} w_{sc}^{-1} & 0 \\ 0 & 1 \end{pmatrix} \int_{\partial D} \left\langle \tilde{f}_{v,(k,\omega)}\left(W_t \circ g, W \circ g\right), n(y) \right\rangle ds(y). \end{aligned}$$

Similar, for the source terms of the $k\omega$-model we conclude using (2.33) and (2.34)

$$\begin{aligned} V_{\hat{D}}\left(Pr_{k,(k,\omega)}\left(\hat{W}_t, \hat{W}\right)\right)(\hat{t}) &= \frac{u_{\text{ref}}^3 L_{\text{ref}}^{m-1}}{w_{sc}} V_D\left(Pr_{k,(k,\omega)}\left(W_t \circ g, W \circ g\right)\right)(t), \\ V_{\hat{D}}\left(De_{k,(k,\omega)}\left(\hat{W}_t, \hat{W}\right)\right)(\hat{t}) &= u_{\text{ref}}^3 L_{\text{ref}}^{m-1} w_{sc} V_D\left(De_{k,(k,\omega)}\left(W_t \circ g, W \circ g\right)\right)(t), \\ V_{\hat{D}}\left(Pr_{\omega,(k,\omega)}\left(\hat{W}_t, \hat{W}\right)\right)(\hat{t}) &= u_{\text{ref}}^2 L_{\text{ref}}^{m-2} V_D\left(Pr_{\omega,(k,\omega)}\left(W_t \circ g, W \circ g\right)\right)(t), \\ V_{\hat{D}}\left(De_{\omega,(k,\omega)}\left(\hat{W}_t, \hat{W}\right)\right)(\hat{t}) &= u_{\text{ref}}^2 L_{\text{ref}}^{m-2} \omega_{sc}^2 V_D\left(De_{\omega,(k,\omega)}\left(W_t \circ g, W \circ g\right)\right)(t). \end{aligned}$$

As a consequence, compared to the system of equations for the mean flow and for the Spalart-Allmaras model, the equivalent nondimensional system of equations for the $k\omega$-model is different than the original dimensional system. The reason is the introduction of the scaling coefficient ω_{sc}. One can also interpret this as a substitution. Instead of solving directly for ω, we solve for $\omega_{sc}\omega$. Naturally, we could also choose $\omega_{sc} = 1$ instead. Nevertheless, this particular choice has a certain reason discussed in Section 3.6. It is important to give a proper scaling of ω near a no-slip wall.

Finally, to close this section we explicitly write down the considered form of the nondimensional system of equations for the $k\omega$-model. To this end we multiply the equation for ω by $1/\omega_{sc}$ to obtain

$$\begin{aligned} & V_D \begin{pmatrix} \omega_{sc}^{-1} Pr_{k,(k,\omega)} - \omega_{sc} De_{k,(k,\omega)} \\ \omega_{sc}^{-1} Pr_{\omega,(k,\omega)} - \omega_{sc} De_{\omega,(k,\omega)} \end{pmatrix} \left(W_t \circ g, W \circ g\right) \\ = \; & \frac{d}{dt} V_D\left(W_t \circ g\right)(t) + R_{c,\partial D,(k,\omega)}\left(W_t \circ g, W \circ g\right)(t) \\ - \; & \begin{pmatrix} \omega_{sc}^{-1} & 0 \\ 0 & \omega_{sc}^{-1} \end{pmatrix} \int_{\partial D} \left\langle \tilde{f}_{v,(k,\omega)}\left(W_t \circ g, W \circ g\right), n(y) \right\rangle ds(y). \end{aligned} \tag{2.53}$$

Equation (2.53) represents the nondimensional system of equations for the $k\omega$-model. Solving for $\omega_{sc}\omega$ we still need to formulate the nondimensional eddy viscosity (2.21),

$$\mu_t\left(W_t, W\right) = \frac{\hat{\rho}}{\rho_{\text{ref}}} \frac{\hat{k}}{k_{\text{ref}}} \frac{\omega_{\text{ref}}}{\hat{\omega}} = \frac{1}{\omega_{sc}} \rho \frac{k}{\omega} = \frac{\sqrt{\gamma} M_\infty L}{Re} \rho \frac{k}{\omega}. \tag{2.54}$$

Within this section the dimensional Reynolds-averaged Navier-Stokes equations given by (2.1a) have been reformulated in nondimensional form by means of the reference states $\rho_{\text{ref}}, p_{\text{ref}}, T_{\text{ref}}$ and L_{ref}. The dimensional equations (2.1a) and nondimensional equations differ only by a constant factor and are therefore reformulated equivalently.

Moreover, also the turbulence flow equation of the Spalart-Allmaras model (2.18) and of the $k\omega$-model (2.22) have been reformulated in nondimensional form.

Remark 2.3.6 *For the rest of this thesis the dimensional physical quantities are replaced by the nondimensional variables and the integral equations (2.1a), (2.18) and (2.22) are considered and implemented in their nondimensional forms (2.45), (2.52) and (2.53).*

2.4 Initial and boundary value problems

So far, we have only stated the integral equations of interest. That is, the mean flow equations (2.45) together with a system of equations describing the required eddy viscosity μ_t, either (2.52) or (2.53). Naturally, for a closed representation we need to formulate a corresponding boundary value problem.

The boundary value problems of interest in this thesis model the motion of a rigid body through a viscous fluid. We formulate this as a flow past an obstacle, where the center of mass is held in place by appropriate forces and the fluid flow past the obstacle tends to a uniform velocity field at large distances from the obstacle. This consideration corresponds to a wind tunnel experiment. Mathematically, the domain of interest is then an exterior region and the boundary value problems are formulated as exterior flow problems.

Due to a missing lack of theoretical understanding pointed out in the introduction in Chapter 1, the definition of boundary values and conditions at infinity are not straightforward. For example, for a complete and closed formulation the decay behavior of infinity for ρ, u, p and additionally even for $\tilde{\nu}$ or k and ω is required, comparable for example to conditions (1.5) and (1.6). Since this is in general not possible we prescribe these values formally. For the representation of the exterior boundary value problems of interest we introduce the formal setting,

$$\begin{aligned} W_\infty &:= (\rho_\infty, \rho_\infty u_\infty, \rho_\infty E_\infty)\,, \\ W_{t,\infty} &:= \tilde{\nu}_\infty, \qquad \text{or} \qquad W_{t,\infty} := (k_\infty, \omega_\infty)\,. \end{aligned}$$

The actual choice of these values for realization is given in Section 3.6. Furthermore, in the sequel let $D \subset \mathbb{R}^m$ be a bounded domain and for the sake of simplicity, we assume that the boundary of ∂D is connected and that ∂D is an orientable submanifold of $\mathbb{R}^m$ of dimension $m-1$.

Though we have stated the RANS equations in their unsteady form, throughout the rest of this thesis we are only interested in approximating a steady state solution. Hence, we formulate the boundary value problems only for the steady state.

Exterior inviscid flow problem:
Find a function $W^\dagger$ *that satisfies the steady Euler equations in* $\mathbb{R}^m \setminus \overline{D}$*, that is*

$$\frac{d}{dt}W^\dagger(x,t) = 0 \qquad \text{for all} \quad x \in \mathbb{R}^m \setminus \overline{D}, \quad t \geq T^\dagger > 0,$$

and satisfies the boundary condition

$$\langle u, n \rangle = 0 \qquad \text{on} \quad \partial D,$$

and $\lim_{|x|\to\infty} W^\dagger(x,t) = W_\infty$ *uniformly for all directions.*

Exterior viscous flow problem:
Find a function W *that satisfies the steady (laminar) Navier-Stokes equations in* $\mathbb{R}^m \setminus \overline{D}$*, that is*

$$\frac{d}{dt}W^\dagger(x,t) = 0 \qquad \text{for all} \quad x \in \mathbb{R}^m \setminus \overline{D}, \quad t \geq T^\dagger > 0,$$

and satisfies the (adiabatic) boundary conditions

$$u = 0 \qquad \text{and} \qquad \frac{\partial T}{\partial n} = 0 \qquad \text{on} \qquad \partial D$$

in the sense of a trace operator, and $\lim_{|x|\to\infty} W^\dagger(x,t) = W_\infty$ *uniformly for all directions.*

Exterior turbulent flow problem:
Find a function W *that satisfies the steady RANS equations in* $\mathbb{R}^m \setminus \overline{D}$*, that is*

$$\frac{d}{dt}W^\dagger(x,t) = 0 \qquad \text{for all} \quad x \in \mathbb{R}^m \setminus \overline{D}, \quad t \geq T^\dagger > 0,$$

and satisfies the (adiabatic) boundary conditions

$$u = 0 \qquad \text{and} \qquad \frac{\partial T}{\partial n} = 0 \qquad \text{on} \qquad \partial D$$

in the sense of a trace operator, and $\lim_{|x|\to\infty} W(x,t) = W_\infty$ *uniformly for all directions. Additionally, find a function* W_t *that either satisfies the equation of the Spalart-Allamaras or the* $k\omega$*-turbulence model in* $\mathbb{R}^m \setminus \overline{D}$*, and satisfies either the boundary conditions*

$$\tilde{\nu} = 0 \qquad \text{or} \qquad (k,\omega) = (0,\infty) \qquad \text{on} \qquad \partial D$$

in the sense that

$$\lim_{h\to 0^+} \omega\left(x - hn\left(x\right)\right) \to \infty, \qquad x \in \partial D, \tag{2.55}$$

and $\lim_{|x|\to\infty} W_t(x,t) = W_{t,\infty}$ *uniformly for all directions.*

For the formulation of the steady state flow problems we point out the following important fact, which are a direct fact of the nondimensional equations (2.45), (2.52) and (2.53):

Remark 2.4.1 *The steady state flow problems formulated above are mathematically fully specified as follows:*

a) The Exterior inviscid flow problem *is specified by the prescription of a Mach number* M_∞*.*

b) The Exterior viscous flow problem *is specified by the prescription of a Mach number* M_∞*, the ratio of the length to the Reynolds number* L/Re *and the constant* C_{suth}*.*

c) The Exterior turbulent flow problem *is specified by the prescription of a Mach number* M_∞*, the ratio of the length of the Reynolds number* L/Re *and the constant* C_{suth}*.*

Remark 2.4.1 might be a bit surprising. Though the original dimensional Euler and Navier-Stokes equations comprise several physical quantities, their mathematically core is restricted to only a few parameters. Therefore, in general care must be taken when one formulates from the original physical problem the mathematical setting and vice-versa.

2.5 Turbulence modeling: An inverse view

To evaluate the formulation and the accuracy of a turbulence model is not an easy task. From our perspective the following four points are a minimum standard one has to consider:

a) The full differential or integral formulation of the model,

b) its exact implementation,

c) a solution algorithm which is able to compute for a given number of degrees of freedom a machine accurate solution and

d) mesh converged results.

As soon as one of these criteria is not satisfied, certain doubts about the assertions made about a turbulence model arise. Unfortunately, throughout the literature about computational fluid dynamics computational details are often hidden and convincing arguments about convergence are also often missing. In particular, information about the actual formulation of boundary conditions as well as the possible impact of certain limitations of variables to stabilize the solution process is often missing. Even when all these criteria are satisfied and there is full evidence about the implementation, a strict conclusion about the accuracy of computed results is hard to obtain. Typical validation measures are the comparison with measurements which also come from a process which is inaccurate. This observation about this complex topic motivates to reconsider the origin of turbulence modeling as a parameter identification problem, typically an inverse problem which is ill-posed.

Inverse problems occur in many branches of science and mathematics. Usually, these problems involve the determination of some model parameters from observed data, as opposed to the problems arising from physical situations where the model parameters or material properties are known. The latter problems are in general *well-posed*. The mathematical term *well-posed problem* stems from a definition given by Hadamard [23]. He called a problem well-posed, if

a) a solution exists,

b) the solution is unique,

c) the solution depends continuously on the data, in some reasonable topology.

Problems that are not well-posed in the sense of Hadamard are termed *ill-posed*. Inverse problems are typically ill-posed. Of the three conditions for a well-posed problem, the condition of stability is most often violated and has our primary interest. This is motivated by the fact that in all applications the data will be measured and therefore perturbed by noise.

Turbulent flow is in general instantaneous and therefore unsteady. The laminar Navier-Stokes equations are a mathematical model to describe turbulent flows. We consider the **exterior viscous flow problem** and denote an integral curve satisfying this initial boundary value problem by W_{lam}.

Many reasons are responsible for why it is not possible to numerically approximate W_{lam} directly. Besides the open problems to prescribe meaningful physical initial values, one severe issue is the numerical effort to carry out a direct numerical simulation for high Reynolds number turbulent flow. The number of mesh points required and the corresponding degrees of freedom is so large that in general such a computation cannot be realized within an appropriate time interval. Hence, the question arising is as follows:

Does there exist an appropriate modification of the laminar Navier-Stokes equations such that the modified equations exhibit three conditions:

a) A steady state solution exists and it is unique.

b) The steady state solution represents important features of W_{lam}.

c) It is possible to compute numerically an approximation to the steady state solution.

Hence, it is our goal to find a modification of the **exterior viscous flow problem** to enforce a unique steady state solution of this problem. The modified flow problem should be such that its steady state solution represents characteristic features of the originally determined integral curve W_{lam}. At this point, we need to identify characteristic features of solutions to a given exterior viscous flow problem. For example, in applications often these are

a) pressure distributions,

b) skin-friction distributions,

c) velocity profiles,

d) the location of separations.

We summarize these characteristics as given data Y. In general these data are measurements and therefore perturbed by noise. Hence, we assume that instead of Y only noisy data Y^δ satisfying

$$\|Y^\delta - Y\| \leq \delta$$

are available. In general, the noise level $\delta \geq 0$ is unknown.

The RANS equations differ from the laminar Navier-Stokes equations by the additional unknown eddy viscosity μ_t. Heuristically speaking, the effect of μ_t can be interpreted as follows: To enforce a steady state solution of the laminar Navier-Stokes equations, a proper weighting of the diffusion terms needs to be incorporated into these equations. It is the mathematical idea that additional artificial weighting of the diffusion terms exacts a steady state solution. Therefore, the laminar viscosity is replaced by the sum of the laminar viscosity and with a function called eddy viscosity (see (2.9)).

It is the goal of turbulence modeling to prescribe the eddy viscosity such that characteristics of interest are represented by a solution to the **exterior turbulent flow problem**. To give the determination of the unknown function μ_t a more general view, we make the following assumption. We assume that $\mu_t(x,t) \geq 0$ and μ_t is constant outside of the ball $B_d := \{x \in \mathbb{R}^m : |x| \leq d\}$. The characteristics of interest are often data defined on ∂D, that is, $Y \in L^2(\partial D)$. Within this notation we may formulate turbulence modeling as an inverse problem. The forward problem might be stated as follows. Given a method to determine the eddy viscosity μ_t, find

a solution of the **exterior turbulent flow problem**. The inverse problem is to reconstruct the eddy viscosity from given data $Y \in L^2(\partial D)$. To formulate the inverse problem consider the operator

$$\begin{aligned} F : D(F) &\rightarrow L^2(\partial D), \\ \mu_t &\mapsto Y, \end{aligned}$$

which maps the eddy viscosity to the corresponding data, for example a pressure distribution. It is important to notice, that this last description of turbulence modeling is in general the way these models are derived. For a small number of test cases measurements are available, and then the model is constructed and calibrated such that it gives good agreement with these measurements.

The interpretation of turbulence modeling as an inverse problem gives rise to several questions, the most severe might be the one of well-posedness. Already the number of turbulence models in the literature which all yield to similar reconstructions of for example the pressure distributions or even skin frictions coefficients suggest non-uniqueness of the problem. Also, a small perturbation in the given pressure distribution may yield to a significantly different solution of the **exterior turbulent flow problem** and therefore also to a significant difference in the eddy viscosity. This suggests that also the condition of continuous dependency on the data is violated. These observations suggest that turbulence modeling by its mathematical nature is an ill-posed problem, that is the reconstruction of an eddy viscosity such that a solution of the **exterior turbulent flow problem** describes data (e.g. measurements). Therefore, the finding of a general ansatz to represent characteristics of turbulent flows for a whole variety of problems using the RANS equations seems to be challenging, and maybe even impossible.

Though the nature of turbulence modeling might be ill-posed, its application is formulated as forward problem. This is due to the fact that for a given problem in general no data Y^δ are available, which are required to formulate the inverse problem. So, unfortunately at this point in time the reconstruction of μ_t requires the solution of additional integral equations. Hence, we only want to make the point that when trying to evaluate results based on a turbulence model, that is for example the comparison of measurements such as pressure distributions with computed values, one should be aware of the fact that in general good agreement cannot be expected due to the mathematical character of the original problem.

Due to the increase in computational power and the ability to realize scale resolving simulation, we want to close this section and chapter with the following considerations for future work.
Assume that a function describing the eddy viscosity $\mu_t \geq 0$ is given. This function may be prescribed by an established turbulence model. Let us denote the steady state solution of the corresponding **exterior turbulent flow problem** by $W^\dagger_{\text{turb}}$.

Furthermore, the associated possibly time dependent solution of the **exterior laminar flow problem** is given by $W^{\dagger}_{\text{lam}}$. Then, we introduce the error induced by the turbulence model within the time interval $[T_0, T_1]$ by

$$err_{\text{turb}}(T_0, T_1) := \int_{T_0}^{T_1} \left\| W^{\dagger}_{\text{lam}}(., t) - W^{\dagger}_{\text{turb}} \right\| dt, \qquad T_1 \geq T_0.$$

Such an error measure does not only include surface data, but it includes the whole field solution. Naturally, a suitable norm needs to be chosen. We can only expect that $err_{\text{turb}}(T_0, T_1)$ can be small if there exists a continuous relationship between the solution $W^{\dagger}_{\text{lam}}$ of the laminar and $W^{\dagger}_{\text{turb}}$ of the Reynolds-averaged Navier-Stokes equations. That is, the change of the solution due to the modification of the set of equations by the eddy viscosity should lead to small differences in the solutions, mathematically speaking

$$\|\mu_t\| < \delta \qquad \Rightarrow \qquad err_{\text{turb}}(T_0, T_1) < \varepsilon.$$

The evaluation of such errors might be helpful to improve the characterization as well as potentials and shortcomings of established models and those developed from scratch.

Chapter 3

Discretization

This chapter is devoted to the description of the spatial approximation of the integral equations (2.45), (2.52) and (2.53). Before we discuss in detail the discretization of the occurring terms, let us start with some preliminary remarks.

The discretization strategy followed throughout this thesis employs a node centered, finite volume space discretization on unstructured meshes. The computational mesh, which is often called dual mesh, is constructed by the primary grid in a preprocessing step. The dual grid forms the control volumes with the unknowns at vertices of the primary grid. Figures 3.1 shows a triangular grid and the generated computational mesh.

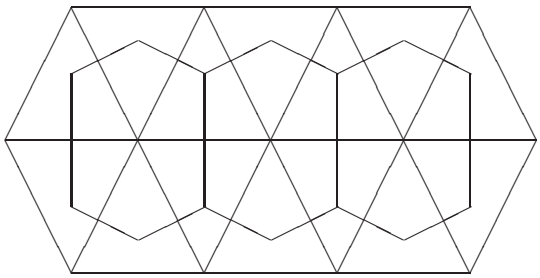

Figure 3.1: Example of a triangular primary grid and its dual grid

For the discretization the distinction between the primary and the dual grid is not necessary. It only emphasizes that the grid which is generated by a mesh generation tool might be different than the actual computational mesh. For the discretization strategy followed in this thesis the required geometric data of the given mesh are:

a) the normal vector for each control surface,

b) the surface area for each control surface,

c) the barycenter for each control volume,

d) the distance to the closest wall for each control volume

Note, technically it does not matter whether this data was generated directly from a given primary mesh or by the introduction of a further intermediate step used to construct a computational mesh. On the other hand, with respect to accuracy, note that solutions obtained on these different computational meshes may differ significantly differ due to the different geometric data. This is in particular true for coarse grids, where an obtained solutions might be far away from being mesh converged. In particular for the formulation of the discretization of boundary conditions the actual mesh geometry has to be considered. The construction of the dual mesh yields so-called half-cells near the boundary, which directly results in an undesired jump in the metric. An illustration of this issue as well as a discussion of the discretization strategies according to this property are given in Section 3.6 and Figure 3.3.

The boundary value problems formulated in Section 2.4 are exterior flow problems. As a consequence, analytically these problems are based on an unbounded domain of definition D. For outer aerodynamic problems, for example consider a free flying aircraft in the atmosphere, this is exactly what needs to be modeled. Naturally, in a discrete sense we cannot deal with an unbounded domain. It needs to be approximated by a bounded computational domain. To satisfy this condition, in general the outer boundary of the computational domain is constructed such that it is a certain number of chord lengths away from the considered body. This distance of the outer boundary to the body may ensure that viscous effects (2.1d) are of negligible effect in the farfield region of the computational domain. Nevertheless, both the restriction to a bounded domain and the necessary formulation of a free stream boundary condition introduce an approximation error which is not classified. Moreover, we will show that the assumption of negligible viscous effects is in general not true (see for example Section 6.10).

We subdivide the description of the discretization of the RANS equations into the inviscid and the viscous part, presented in Sections 3.2 and 3.4, as well as the discretization of the turbulent flow equations including source terms, considered in Section (3.5). Roughly speaking, source terms are all terms which do not belong to the inviscid and viscous terms and cannot be represented using boundary integrals.

Finally note, throughout this section and the next Section 4 the analysis for the discretization strategy and its derivatives is only presented for the three dimensional case $m = 3$. The two dimensional case is significantly simpler.

3.1 Computational mesh and Graph Theory

To discretize the spatial terms of the integral equations we assume that the bounded computational domain is covered by a given finite set of domains $\{D_i\}_{i=1,\ldots,N_{elem}}$. To be more precise we give the following definition.

Definition 3.1.1 *Let $D \subset \mathbb{R}^m$ be a bounded domain. Assume that there exists a finite set of open domains $\{D_i\}_{i=1,\ldots,N_{elem}}$, $D_i \subset \mathbb{R}^m$, $D_i \neq \emptyset$, covering D, that is*

$$D_i \subset D, \qquad \overline{D} = \bigcup_{i=1}^{N_{elem}} \overline{D}_i, \qquad D_i \cap D_j = \emptyset, \quad i \neq j.$$

Then the set

$$M := \{D_i : i = 1, \ldots, N_{elem}\}$$

is called a mesh or a grid or a decomposition covering D.

Definition 3.1.2 *Assume that $a \in \mathbb{R}^m$, $A \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}$.*

a) A hyperplane in $\mathbb{R}^m$ is the set of all points $x \in \mathbb{R}^m$ which satisfy $\langle a, x\rangle = b$.

b) A halfspace is the set of all points $x \in \mathbb{R}^m$ which satisfy $\langle a, x\rangle \leq b$.

c) A polyhedron in $\mathbb{R}^m$ is the intersection of finitely many halfspaces, i.e. the set $\{x : Ax \leq c\}$ for some $c \in \mathbb{R}^n$.

d) A polytope is a bounded polyhedron.

Definition 3.1.3 *Let $D \subset \mathbb{R}^m$, $m = 2, 3$, be a bounded domain and M a decomposition of D. Furthermore assume that D_i, $i = 1, \ldots, N_{elem}$, are polytopes. The decomposition M is called feasible if for all $i \neq j$ either $\overline{D}_i \cap \overline{D}_j = \emptyset$ or one of the following conditions hold:*

a) $\overline{D}_i \cap \overline{D}_j \neq \emptyset$ and $\overline{D}_i$ and $\overline{D}_j$ share exactly one corner, or

b) $\overline{D}_i \cap \overline{D}_j \neq \emptyset$ and $\overline{D}_i$ and $\overline{D}_j$ share exactly one edge, or

c) $\overline{D}_i \cap \overline{D}_j \neq \emptyset$ and $\overline{D}_i$ and $\overline{D}_j$ share exactly one face (m=3).

Definition 3.1.4 *A feasible decomposition M of $D \subset \mathbb{R}^m$ is called a triangulation or in our context a finite volume mesh.*

Definition 3.1.5 *Let $D \subset \mathbb{R}^m$ be a bounded domain.*

a) The volume of D is denoted by

$$\mathrm{vol}\,(D) := \int_D 1 dx.$$

b) The point $x \in \mathbb{R}^m$,

$$x_i := \frac{1}{\mathrm{vol}\,(D)} \int_D y_i dy, \qquad i = 1, \ldots, m,$$

is called the barycenter of D.

There is a close connection between the definition of a mesh and the mathematical structure of a graph. Since we will apply results based on graph theory we shortly introduce the close connection.

Definition 3.1.6 *A graph* $G = (V, E)$ *is a set of vertices* $V = V(G)$ *along with a set of edges* $E = E(G) \subset V \times V$.

Definition 3.1.7 *Let* $G = (V, E)$ *be a graph.*

a) Two vertices in G *are neighbors if and only if they are joined by an edge.*

b) G *is called simple if no vertex is joined to itself by an edge.*

c) G *is finite if the set of vertices is finite and the set of edges is finite.*

d) The degree of a vertex is the number of edges to which it is joined.

e) A path from vertex $v_1 \in V$ *to* $v_n \in V$ *is a sequence of edges*

$$(v_1, v_2), \ldots, (v_{n-1}, v_n), \qquad (v_{i-1}, v_i) \in E, \quad i = 2, \ldots, n.$$

Such a path is a cycle if $v_1, \ldots, v_{n-1}$ *are pairwise distinct and* $v_1 = v_n$.

Definition 3.1.8 *Let* $D \subset \mathbb{R}^m$ *be a bounded domain and assume that* M *is a triangulation of* D. *For two domains* $D_i \in M$ *and* $D_j \in M$, $i \neq j$, *satisfying*

$$\overline{D}_i \cap \overline{D}_j = \partial D_i \cap \partial D_j$$

we define an edge (also called face) in the mesh M *connecting* D_i *with* D_j *by* $e_{ij} := \partial D_i \cap \partial D_j$. *The set of edges (faces) is given by*

$$E(M) := \{e_{ij} : i, j = 1, \ldots, N_{elem}, e_{ij} \neq \emptyset\}$$

Theorem 3.1.9 *Let us denote by* $\Pi_G^M : M \to \mathbb{N}$ *the mapping which projects each subdomain* $D_i \subset M$ *of a mesh* M *to its natural number,*

$$\begin{array}{rcl} \Pi_G^M : M & \to & \{1, \ldots, N_{elem}\} \\ D_i & \mapsto & i, \end{array}$$

and by $\partial\Pi_G^M : E(M) \to \mathbb{N} \times \mathbb{N}$ *the mapping which projects each edge (face) to its tuple of numbers,*

$$\begin{aligned} \partial\Pi_G^M : E(M) &\to \{1, \ldots, N_{elem}\} \times \{1, \ldots, N_{elem}\} \\ e_{ij} &\mapsto i \times j. \end{aligned}$$

Then the mappings Π_G^M *and* $\partial\Pi_G^M$ *are invertible and* $\left(\Pi_G^M(M), \partial\Pi_G^M(E(M))\right)$ *is a graph.*

Proof: First of all note that

$$\Pi_G^M(M) = \{1, \ldots, N_{elem}\}$$

describes a set of vertices and that there exists a one-to-one correspondence between the index i and its corresponding domain D_i. Second, since $e_{ij} \in E(M)$ we have $e_{ij} \neq \emptyset$ and hence $\partial\Pi_G^M(E(M)) \in \Pi_G^M(M) \times \Pi_G^M(M)$. The invertibility of $\partial\Pi_G^M$ follows from the definition of the edges given in Definition 3.1.8.

□

Using the mappings $\Pi_G^M, \partial\Pi_G^M$ and their inverse $\Pi_M^G := \left(\Pi_G^M\right)^{-1}, \partial\Pi_M^G := \left(\partial\Pi_G^M\right)^{-1}$ we can identify the computational mesh M and its corresponding graph $\Pi_G^M(M)$. Hence, in the following we will use this identification straightforward without explicit information.

Definition 3.1.10 *Let* $D \subset \mathbb{R}^m$ *be a bounded domain, assume that* M *is a triangulation of* D, $D_i \in M$.

a) If

$$\overline{D}_i \cap (\mathbb{R}^m \setminus D) = \emptyset$$

we call D_i *an inner domain or an inner element of* M.

b) If

$$\overline{D}_i \cap (\mathbb{R}^m \setminus D) \neq \emptyset$$

we define the edge (face)

$$e_{i,\text{bdry}} := \overline{D}_i \cap (\mathbb{R}^m \setminus D)$$

and call it a boundary edge (face).

c) The number of boundary edges (faces) is denoted by N_{bdry}.

d) The set of boundary edges (faces) is given by

$$E_{\text{bdry}}(M) := \{e_{i,\text{bdry}} : i = 1, \ldots, N_{\text{bdry}}\}.$$

Definition 3.1.11 *Assume that M is a triangulation of D and $D_i \in M$.*

a) The neighbors of a vertex i are denoted by $\mathcal{N}(i)$ and the number of neighbors or the degree of i is given by $\#\mathcal{N}(i)$.

b) The barycenter of D_i is denoted by p_i.

c) Let $e_{ij} \in E(M)$. Then the Euclidean distance of the barycenter p_i of D_i and p_j of D_j is denoted by

$$\operatorname{dist}(e_{ij}) := \|p_i - p_j\|_2. \tag{3.1}$$

Throughout the rest of this thesis we only deal with finite volume meshes. So, in the following, when the word mesh or grid is mentioned, it always means a finite volume mesh or a triangulation.

To define the discretization let us finally define some required geometric data.

Definition 3.1.12 *Assume that M is a triangulation of D and that $e_{ij} \in E(M)$, and $e_{i,\mathrm{bdry}} \in E_{\mathrm{bdry}}(M)$.*

a) The surface area of the face e_{ij} is denoted by

$$\operatorname{svol}(e_{ij}) := \int_{\partial D_i \cap \partial D_j} 1 ds.$$

b) The outer unit normal vector of the face e_{ij} is denoted by

$$n_{e_{ij}} = n_{ij} = (n_{1,ij}, \ldots, n_{m,ij}).$$

c) The surface area of the boundary face $e_{i,\mathrm{bdry}}$ is denoted by

$$\operatorname{svol}(e_{i,\mathrm{bdry}}) := \int_{\partial D_i \cap (\mathbb{R}^m \setminus D)} 1 ds.$$

d) The outer unit normal vector of the face $e_{i,\mathrm{bdry}}$ is denoted by

$$n_{i,\mathrm{bdry}} = (n_{1,i,\mathrm{bdry}}, \ldots, n_{m,i,\mathrm{bdry}}).$$

For polytopes these geometric data can in general be determined using explicit formulae.

3.2 Discretization: The inviscid part

In order to construct a discretization for the inviscid terms (2.1c) we consider a weak form of the Euler equations. To this end, to follow a classic derivation of this method, we shortly mention the classical differential form of the Euler equations. For the incompressible Navier-Stokes equations we already mentioned their differential form in the Introduction (see equations (1.1) and (1.2)). The differential form of the Euler equations is given by

$$\frac{\partial}{\partial t}W(x,t) + \operatorname{div} f_c\left(W(x,t)\right) = 0. \tag{3.2}$$

Introducing a suited test function v to multiplicate (3.2), and using the identity

$$\operatorname{div}\left(v f_c\right) = v \operatorname{div}\left(f_c\right) + \left\langle f_c, \operatorname{grad} v\right\rangle,$$

integration by parts yields the variational formulation

$$\int_D v\frac{dW}{dt}dx - \int_D \left\langle \operatorname{grad} v, f_c(W)\right\rangle dx + \int_{\partial D}\left\langle v f_c(W), n\right\rangle ds = 0. \tag{3.3}$$

Equation (3.3) holds for all possible test functions v. Denoting by $\Pi_k(D_i)$ the set of all polynomials with degree at most k on D_i and consider for a triangulation M covering D the space of functions

$$Sp_k\left(M\right) := \left\{v \in L^2\left(M\right) : v|_{D_i} \in \Pi_k\left(D_i\right), i = 1, \ldots, N_{\text{elem}}\right\}, \tag{3.4}$$

it is the task to find $W_M \in Sp_k\left(M\right)$ such that equation (3.3) carried over to M satisfies

$$\begin{aligned}\sum_{i=1}^{N_{elem}} \Bigg(&\int_{D_i} v_M \frac{dW_M}{dt}dx - \int_{D_i}\left\langle \operatorname{grad} v_M, f_c(W_M)\right\rangle dx \\ + &\int_{\partial D_i}\left\langle v_M f_c(W_M), n\right\rangle ds\Bigg) = 0.\end{aligned} \tag{3.5}$$

This is a classical discontinuous Galerkin ansatz [4]. For the discretization quadrature formulae for the volume and surface integrals are required. Obviously, the evaluation of the surface integrals leads into trouble when the vector field W of conserved variables is discontinuous across e_{ij}. For the solution of theses Riemann problems the concept of a numerical flux function has been introduced. These flux functions are also often called Riemann solver. During the last decades a huge variety of numerical flux functions has been developed [21, 25, 26, 30, 70, 75, 77, 86, 107, 108, 52, 51]. Hence, the surface integral in (3.5) is understood as follows

$$\int_{\partial D_i}\left\langle v_M f_c(W_M), n\right\rangle ds = \sum_{j\in\mathcal{N}(i)}\int_{e_{ij}} v_M^i \mathcal{H}\left(W_M^i, W_M^j, n\right) ds,$$

where $\mathcal{H}$ denotes a numerical flux function and v_M^i the evaluation of the continuous continuation of v_M to the boundary of D_i, that is

$$v_M^i(x) := \lim_{h \to +0} v_M^i \left(x + hn(x) \right). \tag{3.6}$$

Definition 3.2.1 *Assume that M is a triangulation of D, $D_i, D_j \in M$. A mapping $\mathcal{H} : C(D_i) \times C(D_j) \times \mathbb{R}^m \to L^1(\partial D_i \cap \partial D_j)$ is called a numerical flux function if it satisfies*

$$\mathcal{H}(W, W, n)|_{e_{ij}} = \langle f_c(W), n \rangle \tag{3.7}$$

$$\mathcal{H}(U, W, n) = -\mathcal{H}(W, U, -n) \tag{3.8}$$

It is not topic of this thesis to consider general polynomial ansatz functions yielding high order schemes. In this thesis we restrict ourselves to a so-called finite volume discretization, that is we consider the space $Sp_0(M)$. Then, we approximate the unknown function W representing it by a sum of constant ansatz functions. To this end we define the indicator function

$$\mathbb{1}_{D_i}(x) := \begin{cases} 1, & x \in D_i \\ 0, & else \end{cases}$$

and approximate W using the barycenters p_i of D_i by $W_h \in Sp_0(M)$,

$$W(x,t) \approx W_h(x,t), \qquad W_h(x,t) := \sum_{i=1}^{N_{elem}} W_i(t) \mathbb{1}_{D_i}(x). \tag{3.9}$$

In our context the coefficients represent the conservative variables

$$W_i(t) := \left(\rho(p_i,t), (\rho u)(p_i,t), (\rho E)(p_i,t) \right)^T, \quad i = 1, \ldots, N_{\text{elem}}. \tag{3.10}$$

To shorten the notation we define for the rest of the thesis

$$\rho_i := \rho(p_i,t), \quad (\rho u)_i := (\rho u)(p_i,t), \qquad (\rho E)_i := (\rho E)(p_i,t).$$

To derive the finite volume discretization note that the functions $W_i \mathbb{1}_{D_i}$ satisfy

$$\text{grad } \left(W_i \mathbb{1}_{D_i} \right)|_{D_i} = 0, \qquad i = 1, \ldots, N_{\text{elem}}, \tag{3.11}$$

and hence, the Galerkin formulation (3.5) with the only test function

$$v_M(x) = \sum_{i=1}^{N_{elem}} \mathbb{1}_{D_i}(x)$$

simplifies the general ansatz (3.5) to

$$\sum_{i=1}^{N_{elem}} \left(\text{vol}(D_i) \frac{dW_i}{dt} + \int_{\partial D_i} \langle f_c(W_h), n \rangle \, ds \right) = 0, \tag{3.12}$$

and the boundary integral in (3.12) is understood as

$$\int_{\partial D_i} \langle f_c(W_h), n\rangle \, ds = \sum_{j\in\mathcal{N}(i)} \int_{e_{ij}} \mathcal{H}\left(W_L\left(i,\mathcal{N}(i)\right), W_R\left(j,\mathcal{N}(j)\right), n\right) ds,$$

with a suited numerical flux function $\mathcal{H}$. Here the notation $W_L\left(i,\mathcal{N}(i)\right)$ and $W_R\left(j,\mathcal{N}(j)\right)$ suggests that the functions to compute the flux over the face e_{ij} are possibly not only given directly by W_i and W_j, but the stencil is maybe non compact. To be more exact, within this thesis we consider reconstructions of the states as follows,

$$W_L\left(i,\mathcal{N}(i)\right) = W\left(W_i, W_{k,k\in\mathcal{N}(i)}\right), \qquad W_R\left(j,\mathcal{N}(j)\right) = W\left(W_j, W_{k,k\in\mathcal{N}(j)}\right)$$

that is the state W_L depends possibly on the coefficient vector W_i and all the surrounding coefficient vectors W_k, $k\in\mathcal{N}(i)$, and the state W_R depends possibly on the coefficient vector W_j and all the surrounding coefficient vectors W_k, $k\in\mathcal{N}(j)$. Hence, the expression used to approximate the surface integral is understood as

$$\int_{e_{ij}} \left\langle f_c(W_h), n_{e_{ij}}\right\rangle ds \;\approx\; \mathrm{svol}\left(e_{ij}\right) \mathcal{H}\left(W_L\left(i,\mathcal{N}(i)\right), W_R\left(j,\mathcal{N}(j)\right), n_{e_{ij}}\right).$$

The equation (3.12) represents in discretized form the conservation law (2.45). Going back to the general discontinuous Galerkin ansatz (3.5), and rewriting this equation using (3.8) by

$$\begin{aligned} &\sum_{i=1}^{N_{elem}} \left(\int_{D_i} v_M \frac{dW_M}{dt} dx - \int_{D_i} \langle \mathrm{grad}\; v_M, f_c(W_M)\rangle \, dx \right) \\ +\; &\sum_{e_{ij}\in E(M)} \int_{e_{ij}} \left(v_M^i - v_M^j\right) \mathcal{H}\left(W_{h,i}, W_{h,j}, n\right) ds \\ +\; &\sum_{e_{i,\mathrm{bdry}}\in E_{\mathrm{bdry}}(M)} \int_{e_{i,\mathrm{bdry}}} \left(v_M^i - v_M^{i,\mathrm{bdry}}\right) \mathcal{H}\left(W_{h,i}, W_{h,i,\mathrm{bdry}}, n\right) ds = 0, \end{aligned} \tag{3.13}$$

we conclude the following:

Remark 3.2.2 *Since $\left(v_M^i - v_M^j\right)|_{e_{ij}} = 0$ is in general only true for the constant test functions such as $\mathbb{1}_{D_k}, k = i, j$, the discrete conservation property is for the discontinuous Galerkin method only true with respect to these constant test functions. In general, the conservation law does not hold for non-constant test functions $v_M \in Sp_k(M)$.*

Finally, a numerical flux function $\mathcal{H}$ is required. Throughout this thesis we restrict ourselves to a central difference scheme with an added matrix valued artificial

viscosity [98, 92]. On a Cartesian mesh the discretization can be shown to be of second order according to a Taylor series expansion, but, on the other hand, notice that a Cartesian grid is not a necessary requirement for the discrete scheme to be second-order accurate. In general, for an unstructured grid the actual order of the discretization is hard to determine.

To handle shocks, that is discontinuities in the solution, a pressure switch is included into the dissipative part. This reduces in a neighborhood of the shock the scheme to first order. To deal with highly stretched meshes a cell stretching coefficient is included into the scheme. For more details about the construction and stability properties of central difference schemes with an added artificial viscosity we refer to [89, 88], and for the considered scheme below we refer to [63, 42]. To shorten the notation we define

$$W_{\mathcal{N}(i,j)} := \left(W_L\left(i,\mathcal{N}(i)\right), W_R\left(j,\mathcal{N}(j)\right)\right). \tag{3.14}$$

Definition 3.2.3 *Assume that M is a triangulation of D, $e_{ij} \in E(M)$. We define a numerical flux function in normal direction n by*

$$\mathcal{H}\left(W_{\mathcal{N}(i,j)}, n\right) := \frac{1}{2}\left[\left\langle f_c\left(W_i\right), n\right\rangle + \left\langle f_c\left(W_j\right), n\right\rangle\right] - d_{ij}\left(W_{\mathcal{N}(i,j)}, n\right), \tag{3.15}$$

where

$$\begin{aligned} d_{ij}\left(W_{\mathcal{N}(i,j)}, n\right) &:= \frac{1}{2}\mathbf{P}_{ij}^{-1}\left|\mathbf{P}_{ij}\mathbf{A}_{ij}^{\mathrm{Roe}}\right|\left\{\Psi_{ij}\left(W_j - W_i\right)\right. \\ &- \left.\xi s_{ij}\left(W_{\mathcal{N}(i,j)}\right)\left(1-\Psi_{ij}\right)\left(L_j\left(W_{\mathcal{N}(i,j)}\right) - L_i\left(W_{\mathcal{N}(i,j)}\right)\right)\right\}, \\ L_i\left(W_{\mathcal{N}(i,j)}\right) &:= \sum_{j\in\mathcal{N}(i)}\left(W_j - W_i\right), \qquad \mathbf{A}_{ij}^{\mathrm{Roe}} := \frac{\partial\left\langle f_c\left(W_{ij,\mathrm{Roe}}\right), n\right\rangle}{\partial W} \\ \Psi_{ij} &:= \min\{\varepsilon_\psi \Psi_{ij}^*, 1\}, \qquad \Psi_{ij}^* := \frac{(p_j - p_i)^2}{(p_j + p_i)^2}, \quad \varepsilon_\psi := 8. \end{aligned} \tag{3.16}$$

The cell stretching coefficient s_{ij} is based on the absolute value of the largest convective eigenvalue scaled by the surface area and computed by

$$\begin{aligned} \lambda_{i,\mathrm{Roe}} &:= \sum_{j\in\mathcal{N}(i)} \mathrm{svol}\left(e_{ij}\right)\lambda_{ij,\mathrm{Roe}}, \\ \lambda_{ij,\mathrm{Roe}} &:= \left|V_{ij,\mathrm{Roe}}\right| + a_{ij,\mathrm{Roe}}, \quad V_{ij,\mathrm{Roe}} := \left\langle u_{ij,\mathrm{Roe}}, n\right\rangle, \\ s_{ij}\left(W_{\mathcal{N}(i,j)}\right) &:= 1 + 2\frac{\sqrt{z_i}\sqrt{z_j}}{\sqrt{z_i}+\sqrt{z_j}}, \qquad z_i := \frac{\lambda_{i,\mathrm{Roe}} - \mathrm{svol}\left(e_{ij}\right)\lambda_{ij,\mathrm{Roe}}}{\mathrm{svol}\left(e_{ij}\right)\lambda_{ij,\mathrm{Roe}}}. \end{aligned} \tag{3.17}$$

In (3.16) $\mathbf{P}_{ij} := \mathbf{P}_{ij}\left(W_{ij,\mathrm{Roe}}\right) \in \mathbb{R}^{5\times 5}$ is an invertible matrix valued operator.

Definition 3.2.4 *Assume that M is a triangulation of D, $e_{ij} \in E(M)$. We define a numerical flux function in normal direction n corresponding to a first order Roe scheme by*

$$\begin{aligned} \mathcal{H}^{1st,\mathrm{Roe}}(W_i, W_j, n) &:= \frac{1}{2}\left[\langle f_c(W_i), n\rangle + \langle f_c(W_j), n\rangle\right] \\ &- \frac{1}{2}\mathbf{P}_{ij}^{-1}\left|\mathbf{P}_{ij}\mathbf{A}_{ij}^{\mathrm{Roe}}\right|(W_j - W_i). \end{aligned} \tag{3.18}$$

Remark 3.2.5 *The matrix $\mathbf{P}_{ij}$ has been introduced for later use to extend the discretization scheme to the incompressible limit. This is discussed in Section 3.3.1. Throughout this section we assume that $\mathbf{P}_{ij} = \mathbf{I}$.*

Remark 3.2.6 *Due to its importance and according to Definition 3.2.3 we explicitly defined the first order Roe scheme in Definition 3.2.4. This scheme is important to formulate robust multigrid algorithms. And, it cannot be directly derived from the numerical flux function in Definition 3.2.3.*

The superscript 'Roe' means that we use Roe averaged variables [77] to evaluate the corresponding term on the edge (face) ij,

$$\rho_{ij,\mathrm{Roe}} := \sqrt{\rho_i \rho_j}, \tag{3.19a}$$

$$(u_{ij,\mathrm{Roe}})_k := \frac{(u_i)_k\sqrt{\rho_i} + (u_j)_k\sqrt{\rho_j}}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \qquad k = 1, \ldots, m, \tag{3.19b}$$

$$H_{ij,\mathrm{Roe}} := \frac{H_i\sqrt{\rho_i} + H_j\sqrt{\rho_j}}{\sqrt{\rho_i} + \sqrt{\rho_j}}. \tag{3.19c}$$

Applying the equation (2.5), definition (3.19c) is used to compute the square of the corresponding Roe averaged speed of sound to obtain the total energy,

$$a_{ij,\mathrm{Roe}}^2 := (\gamma - 1)\left(H_{ij,\mathrm{Roe}} - \frac{1}{2}\left\|u_{ij,\mathrm{Roe}}\right\|_2^2\right), \tag{3.19d}$$

$$E_{ij,\mathrm{Roe}} := H_{ij,\mathrm{Roe}} - \frac{a_{ij,\mathrm{Roe}}^2}{\gamma}. \tag{3.19e}$$

The construction of the linear operator $\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$ is topic of Section 3.2.3.

To understand the effect of the cell stretching coefficient s_{ij} we consider the near wall boundary layer flow resolved by an anisotropic mesh represented in Figure 3.2. Assuming that the magnitude of the speed of sound is much greater than the streamwise normal velocities in the viscous boundary layer, $a_{ik,\mathrm{Roe}} \gg V_{ik,\mathrm{Roe}}$, that the speed

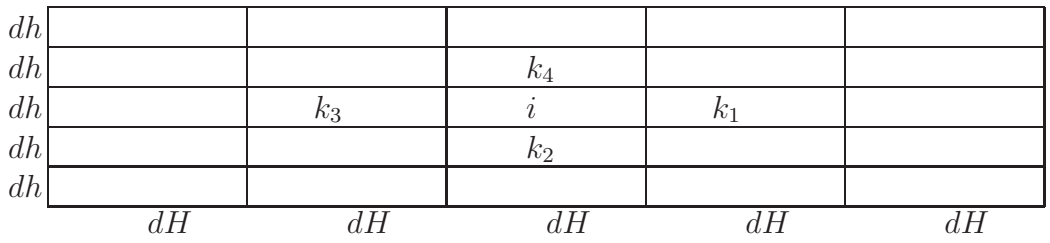

Figure 3.2: Example of an anisotropic Cartesian mesh

of sound is constant over neighboring cells and $dH \gg dh$, we approximate for a vertical face, for example, between the cells i and k_1

$$\begin{aligned} z_i &= \frac{\left(\sum_{k\in\mathcal{N}(i)} \mathrm{svol}\left(e_{ik}\right)\left(|V_{ik,\mathrm{Roe}}| + a_{ik,\mathrm{Roe}}\right)\right) - \mathrm{svol}(e_{ik_1})\left(|V_{ik_1,\mathrm{Roe}}| + a_{ik_1,\mathrm{Roe}}\right)}{\mathrm{svol}(e_{ik_1})\left(|V_{ik_1,\mathrm{Roe}}| + a_{ik_1,\mathrm{Roe}}\right)} \\ &\approx \frac{2dH - dh}{dh} \approx 2\frac{dH}{dh} \end{aligned}$$

and therefore

$$s_{ij}\left(W_{\mathcal{N}(i,j)}\right) \approx 1 + 2\frac{2\frac{dH}{dh}}{2\sqrt{2\frac{dH}{dh}}} = 1 + \sqrt{2}\sqrt{\frac{dH}{dh}}. \tag{3.20}$$

Exchanging the role of dh and dH we get for a horizontal face (e.g. the face between the cells i and k_4

$$z_i \approx \frac{2dH - dH}{dH} \approx 1, \qquad \text{hence} \qquad s_{ij}\left(W_{\mathcal{N}(i,j)}\right) \approx 1 + 2\frac{1}{2} = 2.$$

The approximation (3.20) shows that the factor s_{ij} is designed to increase the dissipation significantly for highly stretched cells in direction of the cell stretching. This is a desired effect to improve the reliability of the discretization. The factor ξ is a global constant weighting factor. In all our computations we choose $\xi = 1/64$.

Theorem 3.2.7 *The numerical flux functions (3.15) and (3.18) satisfy the conditions of Definition 3.2.1.*

Proof: Using the definition of the indicator function $\mathbb{1}_{D_k}$ it is clear that $W_k\mathbb{1}_{D_k}$ represent continuous functions on D_k. Using the continuous continuation (3.6) and the continuity of all other operators, the flux functions yield an integrable function on $\partial D_i \cap \partial D_j$. In case $W_i = W_j$ and $W_L = W_R$ all the differences satisfy $W_j - W_i = 0$. Using (3.6) we conclude (3.7). To prove (3.8) note that

$$\begin{aligned} \frac{1}{2}\left[\langle f_c\left(W_i\right), n\rangle + \langle f_c\left(W_j\right), n\rangle\right] &= -\frac{1}{2}\left[\langle f_c\left(W_j\right), -n\rangle + \langle f_c\left(W_i\right), -n\rangle\right], \\ W_j - W_i &= -(W_i - W_j). \end{aligned}$$

Since $\mathbf{P}_{ij}^{-1}\left|\mathbf{P}_{ij}\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$ and ψ_{ij} are invariant with respect to the change $n \mapsto -n$, we get (3.8).

□

3.2.1 Derivative of convective flux

The implementation of the numerical flux function (3.15) requires knowledge of $\mathbf{A}_{ij}^{\mathrm{Roe}}$ and its eigenvalues and eigenvectors to construct the absolute value of this operator. Although this information can be found in many textbooks (see e.g. [6]), we follow here another way to derive this information. Our way of determining the eigenvalues and eigenvectors does not require rewriting the derivative in primitive variables and convert it back. We end up with a formulation of $\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$, which can be implemented straightforward (see formula (3.33) below).

Theorem 3.2.8 *The derivative of the convective flux $\langle f_c, n\rangle$ in normal direction n is given by*

$$\frac{\partial \langle f_c(W), n\rangle}{\partial W} = V\mathbf{I} + a_1 b_1^T + a_2 b_2^T, \tag{3.21}$$

where

$$\begin{aligned} a_1 &:= (1, u_1, u_2, u_3, H)^T, \qquad a_2 := (0, n_1, n_2, n_3, V)^T, \\ b_1 &:= (-V, n_1, n_2, n_3, 0)^T, \qquad b_2 := \left(\frac{\partial p(W)}{\partial W}\right)^T. \end{aligned}$$

Proof: To prove (3.21) we rewrite the convective part of (2.1a),

$$\langle f_c(W), n\rangle = V\begin{pmatrix}\rho\\ \rho u_1\\ \rho u_2\\ \rho u_3\\ \rho E\end{pmatrix} + p\begin{pmatrix}0\\ n_1\\ n_2\\ n_3\\ 0\end{pmatrix} + Vp\begin{pmatrix}0\\ 0\\ 0\\ 0\\ 1\end{pmatrix} = VW + p\begin{pmatrix}0\\ n_1\\ n_2\\ n_3\\ V\end{pmatrix}.$$

Straightforward differentiation of $\langle f_c, n\rangle$ yields

$$\frac{\partial \langle f_c(W), n\rangle}{\partial W} = V\mathbf{I} + W\frac{\partial V(W)}{\partial W} + \begin{pmatrix}0\\ n_1\\ n_2\\ n_3\\ V\end{pmatrix}\frac{\partial p(W)}{\partial W} + p\begin{pmatrix}0\\ 0\\ 0\\ 0\\ \frac{\partial V(W)}{\partial W}\end{pmatrix}. \tag{3.22}$$

Using (2.2) and the derivative for the normal velocity

$$\frac{\partial V(W)}{\partial W} = \frac{1}{\rho}(-V, n_1, n_2, n_3, 0) \tag{3.23}$$

we simplify (3.22) to

$$\frac{\partial \langle f_c(W), n\rangle}{\partial W} = V\mathbf{I} + \begin{pmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ H \end{pmatrix} (-V, n_1, n_2, n_3, 0) + \begin{pmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ V \end{pmatrix} \frac{\partial p(W)}{\partial W}. \tag{3.24}$$

□

From (3.22) and (3.24) it is obvious that the computation of $\frac{\partial \langle f_c, n\rangle}{\partial W}[W]$ only requires the computation of the derivatives of the normal velocity (3.23) and of the pressure p. Assuming the equation of state (2.3), which we do throughout this thesis, we have

$$\frac{\partial p(W)}{\partial W} = (\gamma - 1)\left(\frac{\|u\|_2^2}{2}, -u_1, -u_2, -u_3, 1\right). \tag{3.25}$$

For the representation of the derivative $\frac{\partial \langle f_c, n\rangle}{\partial W}[W]$ we distinguish between column and row vectors, e.g. $\frac{\partial V(W)}{\partial W} \in \mathbb{R}^{1\times 5}$, whereas $W \in \mathbb{R}^{5\times 1}$. In this notation $W\frac{\partial V(W)}{\partial W} \in \mathbb{R}^{5\times 5}$, as expected.

Equation (3.21) is a compact representation for the derivative of $\langle f_c, n\rangle$. For completeness and implementation issues let us write down the derivative in full matrix notation,

$$\frac{\partial \langle f_c(W), n\rangle}{\partial W} = \begin{pmatrix} 0 & n_1 & n_2 & n_3 & 0 \\ \frac{n_1\zeta_2\|u\|_2^2}{2} - u_1V & n_1\zeta_3u_1 + V & n_2u_1 - n_1\zeta_2u_2 & n_3u_1 - n_1\zeta_2u_3 & n_1\zeta_2 \\ \frac{n_2\zeta_2\|u\|_2^2}{2} - u_2V & n_1u_2 - n_2\zeta_2u_1 & n_2\zeta_3u_2 + V & n_3u_2 - n_2\zeta_2u_3 & n_2\zeta_2 \\ \frac{n_3\zeta_2\|u\|_2^2}{2} - u_3V & n_1u_3 - n_3\zeta_2u_1 & n_2u_3 - n_3\zeta_2u_2 & n_3\zeta_3u_3 + V & n_3\zeta_2 \\ (\zeta_2\|u\|_2^2 - \gamma E)V & n_1\zeta_1 - \zeta_2u_1V & n_2\zeta_1 - \zeta_2u_2V & n_3\zeta_1 - \zeta_2u_3V & \gamma V \end{pmatrix}$$

where

$$\zeta_1 := \gamma E - \Phi, \qquad \zeta_2 := \gamma - 1, \qquad \zeta_3 := 2 - \gamma, \qquad \Phi := \frac{1}{2}(\gamma - 1)\|u\|_2^2.$$

3.2.2 Eigendecomposition of the derivative of the convective flux

The compact representation (3.21) allows a straightforward computation of the eigenvalues and eigenvectors of the derivative of the convective flux.

Theorem 3.2.9 *The derivative matrix of the convective flux given by (3.21) has the following set of eigenpairs:*

$$\{(V, g_1), (V, g_2), (V, g_3), (V + a, g_4), (V - a, g_5)\}$$

where

$$\begin{aligned} g_1 &:= n_1 y_1 + a y_2, \\ g_2 &:= n_2 y_1 + a y_3, \\ g_3 &:= n_3 y_1 + a y_4, \\ g_4 &:= a_1 + a a_2, \\ g_5 &:= a_1 - a a_2. \end{aligned}$$

Here a_1 and a_2 are defined in Theorem 3.2.8 and

$$y_1 := \begin{pmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ \frac{\|u\|_2^2}{2} \end{pmatrix}, \quad y_2 := \begin{pmatrix} 0 \\ 0 \\ n_3 \\ -n_2 \\ u_2 n_3 - u_3 n_2 \end{pmatrix},$$

$$y_3 := \begin{pmatrix} 0 \\ -n_3 \\ 0 \\ n_1 \\ u_3 n_1 - u_1 n_3 \end{pmatrix} \quad \textit{and} \quad y_4 := \begin{pmatrix} 0 \\ n_2 \\ -n_1 \\ 0 \\ u_1 n_2 - u_2 n_1 \end{pmatrix}.$$

Proof: Using representation (3.21) it is obvious that $x \in \mathbb{R}^5$ is eigenvector with eigenvalue V of $\frac{\partial \langle f_c, n \rangle}{\partial W}$ if the following orthogonality relations hold:

$$b_1^T x = b_2^T x = 0 \tag{3.26}$$

It can be easily verified that the vectors $y_1, \ldots, y_4$ satisfy (3.26). However, the vectors y_2, y_3 and y_4 are linearly dependent since

$$n_1 y_2 + n_2 y_3 + n_3 y_4 = 0.$$

Three linear independent eigenvectors may be obtained by g_1, g_2 and g_3. To identify the remaining eigenvectors we use the vectors a_1, a_2 and b_1, b_2 from Theorem 3.2.8 and the relations

$$b_1^T a_1 = b_2^T a_2 = 0, \quad b_1^T a_2 = 1 \quad \text{and} \quad b_2^T a_1 = a^2. \tag{3.27}$$

Then we compute using (3.21)

$$\begin{aligned}\frac{\partial \langle f_c(W), n\rangle}{\partial W}(a_1 + aa_2) &= Va_1 + a^2a_2 + Vaa_2 + aa_1 \\ &= (V+a)(a_1+aa_2), \\ \frac{\partial \langle f_c(W), n\rangle}{\partial W}[W](a_1 - aa_2) &= Va_1 + a^2a_2 - Vaa_2 - aa_1 \\ &= (V-a)(a_1-aa_2).\end{aligned}$$

□

Finally, to implement the numerical flux function $\mathcal{H}$ in (3.15) the inverse of the matrix corresponding to the eigenvectors of $\frac{\partial \langle f_c, n\rangle}{\partial W}$,

$$G := (g_1, g_2, g_3, g_4, g_5) \tag{3.28}$$

is required.

Theorem 3.2.10 *We define the vectors*

$$\begin{aligned} r_1 &:= \frac{\gamma-1}{a^2}(H - \|u\|_2^2, u_1, u_2, u_3, -1)^T, && \text{(3.29a)} \\ r_2 &:= (u_2n_3 - u_3n_2, 0, -n_3, n_2, 0)^T, && \text{(3.29b)} \\ r_3 &:= (u_3n_1 - u_1n_3, n_3, 0, -n_1, 0)^T, && \text{(3.29c)} \\ r_4 &:= (u_1n_2 - u_2n_1, -n_2, n_1, 0, 0)^T. && \text{(3.29d)} \end{aligned}$$

and

$$\begin{aligned} q_1 &:= n_1r_1^T - \frac{1}{a}r_2^T, && \text{(3.30a)} \\ q_2 &:= n_2r_1^T - \frac{1}{a}r_3^T, && \text{(3.30b)} \\ q_3 &:= n_3r_1^T - \frac{1}{a}r_4^T, && \text{(3.30c)} \\ q_4 &:= \frac{1}{2a^2}\left(b_2^T + ab_1^T\right), && \text{(3.30d)} \\ q_5 &:= \frac{1}{2a^2}\left(b_2^T - ab_1^T\right). && \text{(3.30e)} \end{aligned}$$

Then the inverse of G is given by

$$G^{-1} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix}.$$

Proof: Recall the definitions of $y_1, \ldots, y_4$ and $g_1, \ldots, g_5$ given in Theorem 3.2.9 and equation (2.5). To prove the theorem we use the important equations

$$\begin{aligned}
r_1^T y_1 &= \frac{\gamma-1}{a^2}\left(H - \|u\|_2^2 + \|u\|_2^2 - \frac{\|u\|_2^2}{2}\right) = \frac{\gamma-1}{a^2}\left(H - \frac{\|u\|_2^2}{2}\right) = 1,\\
r_1^T y_2 &= r_1^T y_3 = r_1^T y_4 = r_2^T y_1 = r_3^T y_1 = r_4^T y_1 = 0,\\
r_2^T y_2 &= -n_3^2 - n_2^2, \quad r_3^T y_3 = -n_3^2 - n_1^2, \qquad r_4^T y_4 = -n_1^2 - n_2^2,\\
r_2^T y_3 &= n_1 n_2, \qquad r_2^T y_4 = n_1 n_3, \qquad r_3^T y_2 = n_1 n_2,\\
r_3^T y_4 &= n_2 n_3, \qquad r_4^T y_2 = n_1 n_3, \qquad r_4^T y_3 = n_2 n_3,\\
b_1^T y_1 &= b_1^T y_2 = b_1^T y_3 = b_1^T y_4 = 0,\\
b_2^T y_1 &= b_2^T y_2 = b_2^T y_3 = b_2^T y_4 = 0,
\end{aligned}$$

and (3.27) to conclude

$$\begin{aligned}
q_1 g_1 &= (n_1^2 r_1^T y_1 + a n_1 r_1^T y_2 - \frac{n_1}{a} r_2^T y_1 - r_2^T y_2) = (n_1^2 + n_2^2 + n_3^2) = 1,\\
q_1 g_2 &= q_1 g_3 = q_1 g_4 = q_1 g_5 = 0\\
q_2 g_2 &= (n_2^2 r_1^T y_1 + n_2 a r_1^T y_3 - \frac{n_2}{a} r_3^T y_1 - r_3^T y_3) = (n_1^2 + n_2^2 + n_3^2) = 1,\\
q_2 g_1 &= q_2 g_3 = q_2 g_4 = q_2 g_5 = 0\\
q_3 g_3 &= (n_3^2 r_1^T y_1 + n_3 a r_1^T y_4 - \frac{n_3}{a} r_4^T y_1 - r_4^T y_4) = (n_1^2 + n_2^2 + n_3^2) = 1,\\
q_3 g_1 &= q_3 g_2 = q_3 g_4 = q_3 g_5 = 0\\
q_4 g_4 &= \frac{1}{2a^2}(b_2^T a_1 + a b_2^T a_2 + a b_1^T a_1 + a^2 b_1^T a_2) = \frac{1}{2a^2}(a^2 + a^2) = 1,\\
q_4 g_1 &= q_4 g_2 = q_4 g_3 = q_4 g_5 = 0,\\
q_5 g_5 &= \frac{1}{2a^2}(b_2^T a_1 - a b_2^T a_2 - a b_1^T a_1 + a^2 b_1^T a_2) = \frac{1}{2a^2}(a^2 + a^2) = 1,\\
q_5 g_1 &= q_5 g_2 = q_5 g_3 = q_5 g_4 = 0.
\end{aligned}$$

This proves the assertion.

□

3.2.3 Implementation of the Matrix Dissipation operator

Remark 3.2.11 *In this and the following Sections 3.3.2,3.3.3 and 3.3.4 we neglect in the notation the subscript ij. Everything is understood as face values using Roe-averaged variables.*

To implement the matrix valued operator $\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$ required in Definition 3.2.3 note that due to Theorems 3.2.9 and 3.2.10 we have the representation

$$\frac{\partial \left\langle f_c\left(W\right), n\right\rangle}{\partial W} = \sum_{j=1}^{5} \alpha_j g_j q_j, \tag{3.32}$$

where the scalars α_j are given by the eigenvalues

$$\alpha_1 = \alpha_2 = \alpha_3 = V, \qquad \alpha_4 = V + a, \qquad \alpha_5 = V - a.$$

Therefore, we do not follow the standard implementation introduced in [98, 92] and extended to 3-D in [104], but we define the absolute value by the functional calculus

$$\left| \frac{\partial \langle f_c(W), n \rangle}{\partial W} \right| := \sum_{j=1}^{5} |\alpha_j| g_j q_j. \tag{3.33}$$

Formula (3.33) gives the prototype to implement the operator $\left|\mathbf{A}^{\mathrm{Roe}}\right|$ required for the flux function $\mathcal{H}$ in (3.15).

Remark 3.2.12 *A straightforward implementation of this operator in general gives an unstable discretization scheme, in particular when one of the eigenvalues is* ≈ 0*. This happens for example in stagnation points or in the neighborhood of a shock. To avoid instabilities when one of the eigenvalues is* ≈ 0 *in general some so-called entropy fix is required.*

For some $0 \leq \delta_{\mathrm{ef}} \leq 1$ the entropy fix described most often in the literature is to replace the absolute eigenvalues by

$$|\Lambda|_{\mathrm{ef}} := \mathrm{diag}\left(|V|_{\mathrm{ef},1}, |V|_{\mathrm{ef},1}, |V|_{\mathrm{ef},1}, |V+a|_{\mathrm{ef},2}, |V-a|_{\mathrm{ef},3} \right)$$

where

$$|V|_{\mathrm{ef},1} := |\lambda_i|_{\mathrm{ef},1} := \max\left\{ |V|, \delta_{\mathrm{ef}} (|V| + a) \right\}, \qquad i = 1,2,3, \tag{3.34a}$$

$$|V+a|_{\mathrm{ef},2} := |\lambda_4|_{\mathrm{ef},2} := \max\left\{ |V+a|, \delta_{\mathrm{ef}} (|V| + a) \right\}, \tag{3.34b}$$

$$|V-a|_{\mathrm{ef},3} := |\lambda_5|_{\mathrm{ef},3} := \max\left\{ |V-a|, \delta_{\mathrm{ef}} (|V| + a) \right\}. \tag{3.34c}$$

Throughout this thesis we used a value of $\delta_{\mathrm{ef}} = 1/5$. Choosing for example $\delta_{\mathrm{ef}} = 1$ one directly gets a scalar dissipative scheme. Finally, we define

$$\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}} := G\, |\Lambda|_{\mathrm{ef}}\, G^{-1}. \tag{3.35}$$

Another entropy fix is the so-called Harten entropy fix [27]. Here α_j is replaced by

$$\alpha_j^* := \begin{cases} \alpha_j, & \alpha_j > \delta_{\mathrm{H}} \\ \frac{\alpha_j^2 + \delta_{\mathrm{H}}^2}{2\delta^{\mathrm{H}}}, & \alpha_j \leq \delta_{\mathrm{H}}. \end{cases} \tag{3.36}$$

In this formulation $0 < \delta_{\mathrm{H}}$ usually depends on the speed of sound. One often finds the choice $\delta_{\mathrm{H}} := a/4$ or $\delta_{\mathrm{H}} := a/8$. Again, note that for the choice $\delta_{\mathrm{H}} = \alpha_j$ no entropy fix is used.

The combination of (3.33), (3.34) and (3.35) gives the efficient implementation

$$\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}} = \left(\sum_{j=1}^{3} |\lambda_j|_{\mathrm{ef},1}\, g_j q_j\right) + |\lambda_4|_{\mathrm{ef},2}\, g_4 q_4 + |\lambda_5|_{\mathrm{ef},3}\, g_5 q_5. \tag{3.37}$$

Note that the representation (3.37) does not require the construction of the full matrix $\mathbf{A}^{\mathrm{Roe}} \in \mathbb{R}^{5\times 5}$ since only the vector products $q_i x$, $x \in \mathbb{R}^5$ need to be implemented.

Nevertheless, in our implementation we do not use the representation (3.37). Following ideas of Rossow [78, 79] we are going to represent the weighting operator $\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}$ with respect to the local Mach number. Based on this representation we will discuss in Section 3.3 a modification allowing for an extension to incompressible, low Mach number flows.

Theorem 3.2.13 *Assume that* $0 \le \delta_{\mathrm{ef}} \le 1$ *and define*

$$\begin{aligned}
|M|_{\mathrm{ef},1} &:= \max\left\{|M|, \delta_{\mathrm{ef}}\left(|M|+1\right)\right\}, && \text{(3.38a)}\\
|M+1|_{\mathrm{ef},2} &:= \max\left\{|M+1|, \delta_{\mathrm{ef}}\left(|M|+1\right)\right\}, && \text{(3.38b)}\\
|M-1|_{\mathrm{ef},3} &:= \max\left\{|M-1|, \delta_{\mathrm{ef}}\left(|M|+1\right)\right\}, && \text{(3.38c)}\\
M_0^{(1)} &:= \frac{1}{2}\left(|M+1|_{\mathrm{ef},2} - |M-1|_{\mathrm{ef},3}\right), && \text{(3.38d)}\\
M_0^{(2)} &:= \frac{1}{2}\left(-2|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right). && \text{(3.38e)}
\end{aligned}$$

where M denotes the Mach number on the edge e_{ij}, that is

$$M = M_{ij,\mathrm{Roe}} = \frac{V_{ij,\mathrm{Roe}}}{a_{ij,\mathrm{Roe}}}.$$

Note that (3.38) correspond to (3.34) expressed in terms of the Mach number. Then the operator $\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}$ has the following representation:

$$\begin{aligned}
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{1,1} &= |V|_{\mathrm{ef},1} - V M_0^{(1)} + \frac{1}{a}\frac{\|u\|_2^2}{2}(\gamma-1)M_0^{(2)} && \text{(3.39a)}\\
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{1,i+1} &= n_i M_0^{(1)} + \frac{1}{a}(1-\gamma)u_i M_0^{(2)}, \qquad i=1,2,3 && \text{(3.39b)}\\
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{1,5} &= \frac{1}{a}(\gamma-1)M_0^{(2)}, && \text{(3.39c)}
\end{aligned}$$

$$\begin{aligned}
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{i+1,1} &= \frac{\|u\|_2^2}{2}(\gamma-1)\left(\frac{1}{a}u_i M_0^{(2)} + n_i M_0^{(1)}\right) \\
&- V\left(a n_i M_0^{(2)} + u_i M_0^{(1)}\right), \qquad i=1,2,3, \qquad (3.39\mathrm{d}) \\
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{i+1,j+1} &= n_j u_i M_0^{(1)} + \delta_{ij}|V|_{\mathrm{ef},1} + n_i n_j a M_0^{(2)} \\
&+ (1-\gamma)u_j\left(\frac{1}{a}u_i M_0^{(2)} + n_i M_0^{(1)}\right), i,j=1,2,3, \qquad (3.39\mathrm{e}) \\
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{i+1,5} &= (\gamma-1)\left(\frac{1}{a}u_i M_0^{(2)} + n_i M_0^{(1)}\right), \quad i=1,2,3, \qquad (3.39\mathrm{f}) \\
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{5,5} &= |V|_{\mathrm{ef},1} + (\gamma-1)\left(V M_0^{(1)} + \frac{1}{a}H M_0^{(2)}\right) \qquad (3.39\mathrm{g}) \\
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{5,1} &= -V^2 a M_0^{(2)} - V H M_0^{(1)} \\
&+ \frac{\|u\|_2^2}{2}(\gamma-1)\left(V M_0^{(1)} + \frac{1}{a}H M_0^{(2)}\right), \qquad (3.39\mathrm{h}) \\
\left(\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}\right)_{5,i+1} &= (1-\gamma)u_i\left(V M_0^{(1)} + \frac{1}{a}H M_0^{(2)}\right) \\
&+ n_i V a M_0^{(2)} + n_i H M_0^{(1)}, \qquad i=1,2,3. \qquad (3.39\mathrm{i})
\end{aligned}$$

Proof: In principle the representation follows from straightforward computations. We carry out the computation term by term. To this end we introduce another set of primitive variables

$$W_{prim}^{(1)} := (\rho, u_1, u_2, u_3, p).$$

The matrices converting the conservative variables to the primitive variables $W_{prim}^{(1)}$ and back are given by

$$\frac{\partial W}{\partial W_{prim}^{(1)}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ u_1 & \rho & 0 & 0 & 0 \\ u_2 & 0 & \rho & 0 & 0 \\ u_3 & 0 & 0 & \rho & 0 \\ \frac{\|u\|_2^2}{2} & \rho u_1 & \rho u_2 & \rho u_3 & \frac{1}{\gamma-1} \end{pmatrix},$$

$$\frac{\partial W_{prim}^{(1)}}{\partial W} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{u_1}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{u_2}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{u_3}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ (\gamma-1)\frac{\|u\|_2^2}{2} & (1-\gamma)u_1 & (1-\gamma)u_2 & (1-\gamma)u_3 & \gamma-1 \end{pmatrix}.$$

Then, using (3.35) we exploit the following identity:

$$\begin{aligned}\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}} &= \frac{1}{2}\frac{\partial W}{\partial W^{(1)}_{prim}} G^{(1)}_{prim}\left[\Lambda + |\Lambda|_{\mathrm{ef}}\right]\left(G^{(1)}_{prim}\right)^{-1}\frac{\partial W^{(1)}_{prim}}{\partial W} \\ &\quad -\frac{1}{2}\frac{\partial W}{\partial W^{(1)}_{prim}} G^{(1)}_{prim}\left[\Lambda - |\Lambda|_{\mathrm{ef}}\right]\left(G^{(1)}_{prim}\right)^{-1}\frac{\partial W^{(1)}_{prim}}{\partial W}. \end{aligned} \tag{3.40}$$

where G is the matrix of eigenvectors defined in Theorem 3.2.9 and (3.28). The definition of $G^{(1)}_{prim}$ is given by

$$\left(G^{(1)}_{prim}\right) := \frac{\partial W^{(1)}_{prim}}{\partial W} G = \begin{pmatrix} n_1 & n_2 & n_3 & 1 & 1 \\ 0 & -n_3\frac{a}{\rho} & n_2\frac{a}{\rho} & n_1\frac{a}{\rho} & -n_1\frac{a}{\rho} \\ n_3\frac{a}{\rho} & 0 & -n_1\frac{a}{\rho} & n_2\frac{a}{\rho} & -n_2\frac{a}{\rho} \\ -n_2\frac{a}{\rho} & n_1\frac{a}{\rho} & 0 & n_3\frac{a}{\rho} & -n_3\frac{a}{\rho} \\ 0 & 0 & 0 & a^2 & a^2 \end{pmatrix}. \tag{3.41}$$

and its inverse is given by

$$\left(G^{(1)}_{prim}\right)^{-1} := \left(\frac{\partial W^{(1)}_{prim}}{\partial W} G\right)^{-1} = \begin{pmatrix} n_1 & 0 & n_3\frac{\rho}{a} & -n_2\frac{\rho}{a} & -\frac{n_1}{a^2} \\ n_2 & -n_3\frac{\rho}{a} & 0 & n_1\frac{\rho}{a} & -\frac{n_2}{a^2} \\ n_3 & n_2\frac{\rho}{a} & -n_1\frac{\rho}{a} & 0 & -\frac{n_3}{a^2} \\ 0 & \frac{n_1}{2}\frac{\rho}{a} & \frac{n_2}{2}\frac{\rho}{a} & \frac{n_3}{2}\frac{\rho}{a} & \frac{1}{2}\frac{1}{a^2} \\ 0 & -\frac{n_1}{2}\frac{\rho}{a} & -\frac{n_2}{2}\frac{\rho}{a} & -\frac{n_3}{2}\frac{\rho}{a} & \frac{1}{2}\frac{1}{a^2} \end{pmatrix}. \tag{3.42}$$

We now derive an explicit expression for the right hand side of (3.40) using (3.34) and the notation

$$\begin{aligned} \tilde{V}_1 &:= V + |V|_{\mathrm{ef},1} \\ \tilde{V}_2 &:= V + a + |V + a|_{\mathrm{ef},2} \\ \tilde{V}_3 &:= V - a + |V - a|_{\mathrm{ef},3}. \end{aligned}$$

Then we compute

$$\left[\Lambda + |\Lambda|_{\mathrm{ef}}\right]\left(G^{(1)}_{prim}\right)^{-1} = \begin{pmatrix} n_1\tilde{V}_1 & 0 & n_3\frac{\rho\tilde{V}_1}{a} & -n_2\frac{\rho\tilde{V}_1}{a} & -n_1\frac{\tilde{V}_1}{a^2} \\ n_2\tilde{V}_1 & -n_3\frac{\rho\tilde{V}_1}{a} & 0 & n_1\frac{\rho\tilde{V}_1}{a} & -n_2\frac{\tilde{V}_1}{a^2} \\ n_3\tilde{V}_1 & n_2\frac{\rho\tilde{V}_1}{a} & -n_1\frac{\rho\tilde{V}_1}{a} & 0 & -n_3\frac{\tilde{V}_1}{a^2} \\ 0 & \frac{n_1}{2}\frac{\rho\tilde{V}_2}{a} & \frac{n_2}{2}\frac{\rho\tilde{V}_2}{a} & \frac{n_3}{2}\frac{\rho\tilde{V}_2}{a} & \frac{1}{2}\frac{\tilde{V}_2}{a^2} \\ 0 & -\frac{n_1}{2}\frac{\rho\tilde{V}_3}{a} & -\frac{n_2}{2}\frac{\rho\tilde{V}_3}{a} & -\frac{n_3}{2}\frac{\rho\tilde{V}_3}{a} & \frac{1}{2}\frac{\tilde{V}_3}{a^2} \end{pmatrix}.$$

Using the definitions

$$\begin{aligned} \tilde{B}^+ &:= \left(G^{(1)}_{prim}\right)\left[\Lambda + |\Lambda|_{\mathrm{ef}}\right]\left(G^{(1)}_{prim}\right)^{-1}, \\ \tilde{B}^- &:= \left(G^{(1)}_{prim}\right)\left[\Lambda - |\Lambda|_{\mathrm{ef}}\right]\left(G^{(1)}_{prim}\right)^{-1}, \\ \tilde{B} &:= \frac{1}{2}\left(\tilde{B}^+ - \tilde{B}^-\right), \end{aligned}$$

we compute for the first row of $\tilde{B}^+$ term by term

$$\begin{aligned}
\tilde{B}^+_{1,1} &= \sum_{j=1}^{3} n_j n_j \tilde{V}_1 = \tilde{V}_1 \\
\tilde{B}^+_{1,i+1} &= \frac{n_i}{2}\frac{\rho\tilde{V}_2}{a} - \frac{n_i}{2}\frac{\rho\tilde{V}_3}{a} \\
&= \frac{n_i}{2}\left(\frac{\rho\left(V + a + |V+a|_{\mathrm{ef},2}\right)}{a} - \frac{\rho\left(V - a + |V-a|_{\mathrm{ef},3}\right)}{a}\right) \\
&= n_i\rho + \frac{n_i\rho}{2}\left(\left|\frac{V}{a}+1\right|_{\mathrm{ef},2} - \left|\frac{V}{a}-1\right|_{\mathrm{ef},3}\right) \\
&= n_i\rho + \frac{n_i\rho}{2}\left(|M+1|_{\mathrm{ef},2} - |M-1|_{\mathrm{ef},3}\right) \\
&= \frac{n_i\rho}{2}\left(2 + |M+1|_{\mathrm{ef},2} - |M-1|_{\mathrm{ef},3}\right), \qquad i = 1,2,3, \\
\tilde{B}^+_{1,5} &= -\frac{\tilde{V}_1}{a^2} + \frac{1}{2a^2}\left(\tilde{V}_2 + \tilde{V}_3\right) \\
&= \frac{1}{2a^2}\left(-2\left(V + |V|_{\mathrm{ef},1}\right) + V + a + |V+a|_{\mathrm{ef},2} + V - a + |V-a|_{\mathrm{ef},3}\right) \\
&= \frac{1}{2a}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),
\end{aligned}$$

Furthermore, we get

$$\begin{aligned}
\tilde{B}^+_{2,1} &= \tilde{B}^+_{3,1} = \tilde{B}^+_{4,1} = \tilde{B}^+_{5,1} = 0, \\
\tilde{B}^+_{2,2} &= n_3^2\tilde{V}_1 + n_2^2\tilde{V}_1 + \frac{n_1^2}{2}\left(\tilde{V}_2 + \tilde{V}_3\right) \\
&= \tilde{V}_1 - n_1^2\tilde{V}_1 + \frac{n_1^2}{2}\left(2V + |V+a|_{\mathrm{ef},2} + |V-a|_{\mathrm{ef},3}\right) \\
&= \tilde{V}_1 + \frac{n_1^2 a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right), \\
\tilde{B}^+_{3,3} &= \tilde{V}_1 + \frac{n_2^2 a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right), \\
\tilde{B}^+_{4,4} &= \tilde{V}_1 + \frac{n_3^2 a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),
\end{aligned}$$

$$
\begin{aligned}
\tilde{B}^+_{2,3} &= -n_1n_2\tilde{V}_1 + \frac{n_1n_2}{2}\tilde{V}_2 + \frac{n_1n_2}{2}\tilde{V}_3 \\
&= \frac{n_1n_2}{2}\left(-2\tilde{V}_1 + \tilde{V}_2 + \tilde{V}_3\right) \\
&= \frac{n_1n_2a}{2}\left(-2\left|M\right|_{\mathrm{ef},1} + \left|M+1\right|_{\mathrm{ef},2} + \left|M-1\right|_{\mathrm{ef},3}\right), \\
\tilde{B}^+_{3,2} &= \tilde{B}^+_{2,3} \\
&= \frac{n_1n_2a}{2}\left(-2\left|M\right|_{\mathrm{ef},1} + \left|M+1\right|_{\mathrm{ef},2} + \left|M-1\right|_{\mathrm{ef},3}\right), \\
\tilde{B}^+_{2,4} &= -n_1n_3\tilde{V}_1 + \frac{n_1n_3}{2}\tilde{V}_2 + \frac{n_1n_3}{2}\tilde{V}_3 \\
&= \frac{n_1n_3}{2}\left(-2\tilde{V}_1 + \tilde{V}_2 + \tilde{V}_3\right) \\
&= \frac{n_1n_3a}{2}\left(-2\left|M\right|_{\mathrm{ef},1} + \left|M+1\right|_{\mathrm{ef},2} + \left|M-1\right|_{\mathrm{ef},3}\right) \\
\tilde{B}^+_{4,2} &= \tilde{B}^+_{2,4} \\
\tilde{B}^+_{3,4} &= -n_2n_3\tilde{V}_1 + \frac{n_2n_3}{2}\tilde{V}_2 + \frac{n_2n_3}{2}\tilde{V}_3 \\
&= \frac{n_2n_3}{2}\left(-2\tilde{V}_1 + \tilde{V}_2 + \tilde{V}_3\right) \\
&= \frac{n_2n_3a}{2}\left(-2\left|M\right|_{\mathrm{ef},1} + \left|M+1\right|_{\mathrm{ef},2} + \left|M-1\right|_{\mathrm{ef},3}\right)
\end{aligned}
$$

$$
\begin{aligned}
\tilde{B}^+_{4,3} &= \tilde{B}^+_{3,4} \\
\tilde{B}^+_{2,5} &= \frac{n_1}{2A\rho a}\left(\tilde{V}_2 - \tilde{V}_3\right) \\
&= \frac{n_1}{2\rho}\left(2 + \left|M+1\right|_{\mathrm{ef},2} - \left|M-1\right|_{\mathrm{ef},3}\right), \\
\tilde{B}^+_{3,5} &= \frac{n_2}{2\rho}\left(2 + \left|M+1\right|_{\mathrm{ef},2} - \left|M-1\right|_{\mathrm{ef},3}\right), \\
\tilde{B}^+_{4,5} &= \frac{n_3}{2\rho}\left(2 + \left|M+1\right|_{\mathrm{ef},2} - \left|M-1\right|_{\mathrm{ef},3}\right).
\end{aligned}
$$

Finally, the missing terms are computed by

$$\begin{aligned}
\tilde{B}^+_{5,2} &= \frac{n_1 a\rho}{2}\left(\tilde{V}_2 - \tilde{V}_3\right)\\
&= \frac{n_1 a\rho}{2}\left(V + a + |V+a|_{\mathrm{ef},2} - (V - a + |V-a|_{\mathrm{ef},3})\right)\\
&= \frac{n_1 a^2\rho}{2a}\left(2a + |V+a|_{\mathrm{ef},2} - |V-a|_{\mathrm{ef},3}\right)\\
&= \frac{n_1\gamma p}{2}\left(2 + |M+1|_{\mathrm{ef},2} - |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^+_{5,3} &= \frac{n_2\gamma p}{2}\left(2 + |M+1|_{\mathrm{ef},2} - |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^+_{5,4} &= \frac{n_3\gamma p}{2}\left(2 + |M+1|_{\mathrm{ef},2} - |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^+_{5,5} &= \tilde{V}_1 + \frac{a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right).
\end{aligned}$$

Similarly, one obtains

$$\begin{aligned}
\tilde{B}^-_{1,1} &= V - |V|_{\mathrm{ef},1}\\
\tilde{B}^-_{1,i+1} &= \frac{n_i\rho}{2}\left(2 - |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right), \qquad i = 1,2,3,\\
\tilde{B}^-_{1,5} &= \frac{1}{2a}\left(2\,|M|_{\mathrm{ef},1} - |M+1|_{\mathrm{ef},2} - |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{2,1} &= \tilde{B}^-_{3,1} = \tilde{B}^-_{4,1} = \tilde{B}^-_{5,1} = 0,\\
\tilde{B}^-_{2,2} &= V - |V|_{\mathrm{ef},1} - \frac{n_1^2 a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{3,3} &= V - |V|_{\mathrm{ef},1} - \frac{n_2^2 a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{4,4} &= V - |V|_{\mathrm{ef},1} - \frac{n_3^2 a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{2,3} &= -\frac{n_1 n_2}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{3,2} &= \tilde{B}^-_{2,3},\\
\tilde{B}^-_{2,4} &= -\frac{n_1 n_3 a}{2}\left(-2\,|M|_{\mathrm{ef},1} + |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{4,2} &= \tilde{B}^-_{2,4},\\
\tilde{B}^-_{2,5} &= \frac{n_1}{2\rho}\left(2 - |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{3,5} &= \frac{n_2}{2\rho}\left(2 - |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{4,5} &= \frac{n_3}{2\rho}\left(2 - |M+1|_{\mathrm{ef},2} + |M-1|_{\mathrm{ef},3}\right).
\end{aligned}$$

Finally, the last row is given by

$$\begin{aligned}
\tilde{B}^-_{5,2} &= \frac{n_1\gamma p}{2}\left(2-|M+1|_{\mathrm{ef},2}+|M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{5,3} &= \frac{n_2\gamma p}{2}\left(2-|M+1|_{\mathrm{ef},2}+|M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{5,4} &= \frac{n_3\gamma p}{2}\left(2-|M+1|_{\mathrm{ef},2}+|M-1|_{\mathrm{ef},3}\right),\\
\tilde{B}^-_{5,5} &= V-|V|_{\mathrm{ef},1}-\frac{a}{2}\left(-2\,|M|_{\mathrm{ef},1}+|M+1|_{\mathrm{ef},2}+|M-1|_{\mathrm{ef},3}\right).
\end{aligned}$$

Using the definitions (3.38d) and (3.38e) we get

$$\tilde{B}=\begin{pmatrix}
|V|_{\mathrm{ef},1} & n_1\rho M_0^{(1)} & n_2\rho M_0^{(1)} & n_3\rho M_0^{(1)} & \frac{1}{a}M_0^{(2)}\\
0 & |V|_{\mathrm{ef},1}+n_1^2 aM_0^{(2)} & n_1n_2aM_0^{(2)} & n_1n_3aM_0^{(2)} & \frac{n_1}{\rho}M_0^{(1)}\\
0 & n_1n_2aM_0^{(2)} & |V|_{\mathrm{ef},1}+n_2^2aM_0^{(2)} & n_2n_3aM_0^{(2)} & \frac{n_2}{\rho}M_0^{(1)}\\
0 & n_1n_3aM_0^{(2)} & n_2n_3aM_0^{(2)} & |V|_{\mathrm{ef},1}+n_3^2aM_0^{(2)} & \frac{n_3}{\rho}M_0^{(1)}\\
0 & n_1\gamma pM_0^{(1)} & n_2\gamma pM_0^{(1)} & n_3\gamma pM_0^{(1)} & |V|_{\mathrm{ef},1}+aM_0^{(2)}
\end{pmatrix}.$$

To rewrite the representation $\tilde{B}$ into conservative variables we need to evaluate

$$\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}=\frac{\partial W}{\partial W^{(1)}_{prim}}\tilde{B}\frac{\partial W^{(1)}_{prim}}{\partial W}. \tag{3.43}$$

To this end we compute in a first step

$$\begin{aligned}
&\frac{\partial W}{\partial W^{(1)}_{prim}}\tilde{B}\\
&=\begin{pmatrix}
|V|_{\mathrm{ef},1} & n_1\rho M_0^{(1)} & n_2\rho M_0^{(1)} & n_3\rho M_0^{(1)} & \frac{1}{a}M_0^{(2)}\\
u_1|V|_{\mathrm{ef},1} & \zeta_{11} & \zeta_{12} & \zeta_{13} & \frac{u_1}{a}M_0^{(2)}+n_1M_0^{(1)}\\
u_2|V|_{\mathrm{ef},1} & \zeta_{21} & \zeta_{22} & \zeta_{23} & \frac{u_2}{a}M_0^{(2)}+n_2M_0^{(1)}\\
u_3|V|_{\mathrm{ef},1} & \zeta_{31} & \zeta_{32} & \zeta_{33} & \frac{u_3}{a}M_0^{(2)}+n_3M_0^{(1)}\\
\frac{\|u\|_2^2}{2}|V|_{\mathrm{ef},1} & \xi_1 & \xi_2 & \xi_3 & \xi_4
\end{pmatrix}
\end{aligned} \tag{3.44}$$

where

$$\begin{aligned}
\zeta_{i,j} &:= n_j\rho u_iM_0^{(1)}+\rho\left(\delta_{ij}|V|_{\mathrm{ef},1}+n_in_jaM_0^{(2)}\right), \qquad i,j=1,2,3, && \text{(3.45a)}\\
\xi_i &:= \rho u_i|V|_{\mathrm{ef},1}+n_iV\rho aM_0^{(2)}+n_i\rho HM_0^{(1)}, \qquad i=1,2,3, && \text{(3.45b)}\\
\xi_4 &:= \frac{1}{\gamma-1}|V|_{\mathrm{ef},1}+VM_0^{(1)}+\frac{1}{a}HM_0^{(2)}. && \text{(3.45c)}
\end{aligned}$$

Here the equations for $\xi_1, \ldots, \xi_4$ follow by

$$\begin{aligned}
\xi_1 &= n_1\rho\frac{\|u\|_2^2}{2}M_0^{(1)} + \rho u_1\left(|V|_{\mathrm{ef},1} + n_1^2 a M_0^{(2)}\right) + \rho u_2 n_1 n_2 a M_0^{(2)} \\
&\quad + \rho u_3 n_1 n_3 a M_0^{(2)} + \frac{1}{\gamma-1}n_1\gamma p M_0^{(1)} \\
&= \rho u_1|V|_{\mathrm{ef},1} + n_1 V\rho a M_0^{(2)} + n_1\rho M_0^{(1)}\left(\frac{\gamma p}{\rho(\gamma-1)} + \frac{\|u\|_2^2}{2}\right), \\
\xi_4 &= \frac{1}{a}\frac{\|u\|_2^2}{2}M_0^{(2)} + V M_0^{(1)} + \frac{1}{\gamma-1}\left(|V|_{\mathrm{ef},1} + a M_0^{(2)}\right) \\
&= \frac{1}{\gamma-1}|V|_{\mathrm{ef},1} + V M_0^{(1)} + \frac{1}{a}M_0^{(2)}\left(\frac{a^2}{\gamma-1} + \frac{\|u\|_2^2}{2}\right) \\
&= \frac{1}{\gamma-1}|V|_{\mathrm{ef},1} + V M_0^{(1)} + \frac{1}{a}M_0^{(2)} H.
\end{aligned}$$

To obtain the equations above we exploited identity (2.5). For a closed representation of the Roe matrix $\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}$, we finally have to multiply the matrix (3.44) from the right by $\frac{\partial W_{prim}^{(1)}}{\partial W}$. This computation gives the representation (3.39).

□

We show that the computation above correspond one-to-one to the formulae given in [78, 80, 93].

Lemma 3.2.14 *Assume that $\delta_{\mathrm{ef}} = 0$ in (3.34) and (3.38) respectively, that is no entropy fix is applied. Then we have*

$$M_0^{(1)} = M_0, \tag{3.46a}$$
$$M_0^{(2)} = 1 - |M_0| \tag{3.46b}$$

where

$$M_0 := \mathrm{sign}(M)\min\{|M|, 1\}.$$

Proof: Assuming that $\delta_{\mathrm{ef}} = 0$ we neglect in all terms the entropy fix. Then, using

$$|M+1| + |M-1| = \begin{cases} (M+1) + (M-1) = 2M, & M \geq 1 \\ (M+1) + (-1)(M-1) = 2, & -1 \leq M \leq 1 \\ (-1)(M+1) + (-1)(M-1) = -2M, & M \leq -1 \end{cases}$$

we get

$$\begin{aligned}
-2|M| + |M+1| + |M-1| &= \begin{cases} 0, & M \geq 1 \\ -2M+2, & 0 \leq M \leq 1 \\ 2M+2, & -1 \leq M \leq 0 \\ 0, & M \leq -1 \end{cases} \\
&= 2\left(1 - |\mathrm{sign}(M)\min\{|M|, 1\}|\right).
\end{aligned}$$

This proves the equation for $M_0^{(2)}$. The assertion for $M_0^{(1)}$ follows by

$$\begin{aligned} |M+1|-|M-1| &= \begin{cases} (M+1)-(M-1)=2, & M\geq 1, \\ (M+1)-(-1)(M-1)=2M, & -1\leq M\leq 1, \\ -(M+1)-(-1)(M-1)=-2, & M\leq -1. \end{cases} \\ &= 2\,\mathrm{sign}(M)\min\{|M|,1\}. \end{aligned}$$

□

Corollary 3.2.15 *Assume that $\delta_{\mathrm{ef}}=0$ in (3.34) and (3.38) respectively, that is no entropy fix is applied. Then we have*

$$\frac{\partial \mathbf{W}}{\partial \mathbf{W}_{prim}^{(1)}}\tilde{B} = \begin{pmatrix} |V| & n_1\rho M_0 & n_2\rho M_0 & n_3\rho M_0 & \frac{1}{a}(1-|M_0|) \\ u_1|V| & \zeta_{11} & \zeta_{12} & \zeta_{13} & \frac{u_1}{a}(1-|M_0|)+n_1M_0 \\ u_2|V| & \zeta_{21} & \zeta_{22} & \zeta_{23} & \frac{u_2}{a}(1-|M_0|)+n_2M_0 \\ u_3|V| & \zeta_{31} & \zeta_{32} & \zeta_{33} & \frac{u_3}{a}(1-|M_0|)+n_3M_0 \\ \frac{\|u\|_2^2}{2}|V| & \xi_1 & \xi_2 & \xi_3 & \xi_4 \end{pmatrix}$$

where

$$\begin{aligned} \zeta_{i,j} &= n_j\rho u_i M_0+\rho\left(\delta_{ij}|V|+n_in_ja\left(1-|M_0|\right)\right), \qquad i,j=1,2,3, \quad (3.47a) \\ \xi_i &= \rho u_i|V|+n_iV\rho a\left(1-|M_0|\right)+n_i\rho H M_0, \qquad i=1,2,3, \quad (3.47b) \\ \xi_4 &= \frac{1}{\gamma-1}|V|+VM_0+\frac{1}{a}H\left(1-|M_0|\right). \quad (3.47c) \end{aligned}$$

The matrix (3.44) corresponds exactly to the matrix (7.2) in [93].

Proof: The Corollary follows by inserting (3.46) into matrix (3.44). Then one evaluates the expressions (3.45) to conclude (3.47).

□

Again note that the representations (3.39) and (3.37) of $\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}$ are equivalent. For all the numerical computations shown in Section 6 we have used (3.39). And moreover, representation (3.39) allows to identify critical entries of $\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}$ which are responsible for a loss of accuracy in the low Mach number regime. Based on these observations a direct manipulation of terms in $\left|\mathbf{A}^{\mathrm{Roe}}\right|_{\mathrm{ef}}$ can be done to improve the accuracy of the numerical flux $\mathcal{H}$ in (3.15) to the incompressible limit.

3.3 Low Mach number modifications

Based on the Mach number flow fields in fluid mechanics are subdivided into compressible and incompressible flows. Compressible fluid flow is modeled by the Euler or the Navier-Stokes equations and the pressure is given by the equation of

state (2.3). Within the incompressible equations the pressure takes the role of a Lagrangian multiplier such that the velocity field satisfies an elliptic divergence constraint. The incompressible and compressible equations are of different type, but there is no exact dividing line in the sense that there exists a strict criterion to decide whether a flow can be characterized as incompressible or compressible.

Flow solvers based on the compressible equations are in general not suitable to simulate incompressible flow fields and flow fields of varying type. For example, at low Mach number, the Roe scheme [77] as well as other upwind schemes produce an excess of artificial viscosity. A modification of the Roe scheme was presented by Turkel [102, 99] leading to a significant accuracy improvement observed for approximate solutions of the Euler and Navier-Stokes equations. Subsequently, the problem to extend a computer code originally implemented to approximate solutions of the compressible Euler, Navier-Stokes and Reynolds-averaged Navier-Stokes equations has been investigated by many authors.

The principle problem is given by the basic observation that for low-speed flow the system of equations is "stiff", since the ratio of the convective speed to the speed of sound is very small [103]. Since the word "stiff" is in general not a well defined expression, a more detailed approach is an asymptotic analysis of the compressible Euler equations. Based on the characteristic Mach number such a procedure was suggested by Klainermann and Majda [35, 36] exploiting several assumptions on the given fluid. This analysis was improved and generalized by Asano [2] and Ukai [106]. An overview and summary of this procedure was given by Meister [58].

Preconditioning the governing equations was originally suggested by Chorin [11] and afterwards extended and systematically investigated by Turkel [99]. The modification suggested by Turkel is twofold and can be considered as an improvement with respect to accuracy and an acceleration for the solution method (introducing artificial time derivatives which allow for a faster convergence to steady state). Originally the modification was considered as a preconditioning technique which dealt with the acceleration towards the steady state only [97]. Later it was realized that there is also an accuracy problem [102]. Hence, the acceleration of the solution method and the accuracy problem are in principle independent of each other. On the other hand, considering an explicit time marching method, it turns out that the acceleration is only a consequence of the change in the largest eigenvalue of the local linearized operator which is a consequence of the modification of the considered upwind scheme. This consideration sheds light on the fact that the so-called "low Mach number preconditioning" is not an acceleration method, which is possibly implied by the word "preconditioning", but it is a modification of the discretization. Thus, in this context the word "preconditioning" is misleading. But the nomenclature was built on the original acceleration [97]. Nevertheless, in general mathematically the word "preconditioning" describes an equivalent reformulation of the equation to solve. But a change of the discretization also changes

the solution of the discretized system, which means that the "low Mach preconditioned" discretization is not equivalent to the solution of the original discretized equation. So, the word "modification" seems to be more appropriate.

In the report of Viozat [22] an analysis of the low Mach modified scheme can be found. Here, it is shown that the discretization error of a pure Roe scheme depends on the ratio between the mesh size parameter and Mach number, but the modification of Turkel depends only on the mesh size. This is the reason why for a given grid, that is a given mesh size, and low Mach number the modification of Turkel shows a significant accuracy improvement. A similar analysis is also given in [115], and a low Mach modification was implemented in the context of a finite element method. Again, the acceleration of the solution method is only a by-product, which may only be true when one considers explicit time stepping schemes.

Having identified the source of disturbance when simulating low-speed flows using a compressible code, several modifications depending on many parameters have been suggested throughout the literature to significantly improve the accuracy [97, 10, 14, 102, 103, 100]. On the other hand, throughout the literature one rarely finds results where these developed techniques are applied to large scale high Reynolds number turbulent flows, in particular not for unstructured codes. And when results are presented [105, 101], one observes for flows, which have strong local compressible effects, convergence problems. It seems, to the author's point of view, though there is no obvious proof and observation, that the considered methods work well for globally inviscid incompressible fluid flows. For transonic flows and flows with locally strong compressible effects the reliability and robustness of the suggested modifications show a severe deterioration, for example the computation diverges or the finding of a suitable set of parameters is difficult or even impossible. In particular, the last category of flows is of major importance for external aerodynamics, since a typical class of test cases satisfying such a behavior are airfoils and aircraft at high angle of attack (i.e. an aircraft in landing configuration). The complexity of these kinds of flows is already a challenge in itself and not well understood. The mixture of compressible and incompressible effects makes a simulation of such flows even harder.

A similar, but slightly different approach to deal with incompressible flows was suggested by Rossow [78]. Contrary to the other suggested methods, Rossow observed that only some terms in the weighting operator of the artificial viscosity are responsible for the accuracy deterioration. Thus, he exchanged the global weighting of the artificial viscosity with some so-called low Mach number preconditioner by modifying only certain terms. To this end, in a first step the Roe matrix weighting the artificial viscosity is represented equivalently in terms of the Mach number and speed of sound. Then, for low speed flows the speed of sound and convective velocity are modified. In the following, this technique was improved by Swanson, Rossow

and Turkel and incorporated into an implicit time stepping scheme [80, 93, 91]. This method was successfully applied in a structured grid code to globally incompressible flows, with the addition that transonic airfoil flow results were shown. Peles, Turkel and Yaniv demonstrated the applicability of this method with respect to the $k\omega$-SST turbulence model and flows with chemical reactions [72]. Though the operators were given explicitly in the published articles, unfortunately a detailed derivation as well as detailed information with respect to the incorporation of entropy fixes to avoid zero dissipation were missing.

To understand the loss of accuracy of we again refer to [58]. In this section we present two different methods to extend the discretization for the compressible Reynolds averaged Navier-Stokes equations to the incompressible limit.

3.3.1 Modification of Turkel

To care about incompressible effects the discretization of flux terms in (2.45) need to be modified. Hence, we now introduce for the numerical flux function $\mathcal{H}$ given in (3.15) a matrix valued operator $\mathbf{P}_{ij} \neq \mathbf{I}$. To clarify that this modification is introduced for low Mach number flows, we denote this by

$$\mathcal{H}_{\mathrm{LM}}\left(W_{\mathcal{N}(i,j)}, n\right) := \frac{1}{2}\left[\langle f_c\left(W_i\right), n\rangle + \langle f_c\left(W_j\right), n\rangle\right] - d_{ij,\mathrm{LM}}\left(W_{\mathcal{N}(i,j)}\right), \tag{3.48}$$

where

$$\begin{aligned} d_{ij,\mathrm{LM}}\left(W_{\mathcal{N}(i,j)}\right) &:= \frac{1}{2}\mathbf{P}_{ij}^{-1}\left|\mathbf{P}_{ij}\mathbf{A}_{ij}^{\mathrm{Roe}}\right|\left\{\Psi_{ij}\left(W_j - W_i\right)\right. \\ &- \left.\xi s_{ij}\left(W_{\mathcal{N}(i,j)}\right)\left(1-\Psi_{ij}\right)\left(L_j\left(W_{\mathcal{N}(i,j)}\right) - L_i\left(W_{\mathcal{N}(i,j)}\right)\right)\right\}. \end{aligned} \tag{3.49}$$

We now assume that $\mathbf{P}_{ij} \neq \mathbf{I}$. Then, in general the discretization is different compared to $\mathbf{P}_{ij} = \mathbf{I}$, and on a given mesh we expect different results. On the other hand, assuming that for the mesh spacing $h \to 0$ the difference terms satisfy

$$\left(W_j^h - W_i^h\right) \to 0 \qquad \text{and} \qquad \left(L_j(W^h) - L_i(W^h)\right) \to 0,$$

the solutions of the two distinct discrete systems of equations converge to the same limit value if the solution is smooth.

3.3.2 Definition of a low Mach number modification

The generalized numerical flux function $\mathcal{H}_{\mathrm{LM}}$ given in (3.48) is exploited to introduce a modification to extend the discretization strategy to incompressible flows. To this end we introduce a set of primitive variables, the so-called entropy variables

$$W_{prim}^{(0)} := (p, u_1, u_2, u_3, S). \tag{3.50}$$

Here S denotes the entropy determined by $S = \ln(p/\rho^\gamma)$. The operators converting the entropy variables into conservative variables are given by

$$\frac{\partial W^{(0)}_{prim}}{\partial W} = \begin{pmatrix} \frac{\gamma-1}{2}\|u\|_2^2 & (1-\gamma)u_1 & (1-\gamma)u_2 & (1-\gamma)u_3 & \gamma-1 \\ -\frac{u_1}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{u_2}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{u_3}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \frac{\gamma-1}{2}\|u\|_2^2 - a^2 & (1-\gamma)u_1 & (1-\gamma)u_2 & (1-\gamma)u_3 & \gamma-1 \end{pmatrix},$$

and

$$\frac{\partial W}{\partial W^{(0)}_{prim}} = \begin{pmatrix} \frac{1}{a^2} & 0 & 0 & 0 & -\frac{1}{a^2} \\ \frac{u_1}{a^2} & \rho & 0 & 0 & -\frac{u_1}{a^2} \\ \frac{u_2}{a^2} & 0 & \rho & 0 & -\frac{u_2}{a^2} \\ \frac{u_3}{a^2} & 0 & 0 & \rho & -\frac{u_3}{a^2} \\ \frac{1}{\gamma-1} + \frac{M^2}{2} & \rho u_1 & \rho u_2 & \rho u_3 & -\frac{M^2}{2} \end{pmatrix}.$$

The derivative of the convective flux represented with respect to entropy variables (3.50) is given by

$$\frac{\partial \langle f_c, n\rangle}{\partial W^{(0)}_{prim}}\left[W^{(0)}_{prim}\right] = \begin{pmatrix} V & n_1\rho a^2 & n_2\rho a^2 & n_3\rho a^2 & 0 \\ \frac{n_1}{\rho} & V & 0 & 0 & 0 \\ \frac{n_2}{\rho} & 0 & V & 0 & 0 \\ \frac{n_3}{\rho} & 0 & 0 & V & 0 \\ 0 & 0 & 0 & 0 & V \end{pmatrix}.$$

Now, to exploit generalization (3.49) Turkel suggested a parameter dependent family of operators (see e.g.[103]). With respect to entropy variables $W^{(0)}_{prim}$ (3.50) the operator and its inverse are given by

$$\mathbf{P}^{(0)}_{\mathbf{LM}} := \begin{pmatrix} \frac{\beta M_{\text{art}}}{a^2} & 0 & 0 & 0 & -\frac{\beta M_{\text{art}}}{a^2}\delta \\ -\frac{\alpha u_1}{\rho a^2} & 1 & 0 & 0 & \frac{\alpha u_1}{\rho a^2}\delta \\ -\frac{\alpha u_2}{\rho a^2} & 0 & 1 & 0 & \frac{\alpha u_2}{\rho a^2}\delta \\ -\frac{\alpha u_3}{\rho a^2} & 0 & 0 & 1 & \frac{\alpha u_3}{\rho a^2}\delta \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.51}$$

$$\left[\mathbf{P}^{(0)}_{\mathbf{LM}}\right]^{-1} = \begin{pmatrix} \frac{a^2}{\beta M_{\text{art}}} & 0 & 0 & 0 & \delta \\ \frac{\alpha u_1}{\beta\rho M_{\text{art}}} & 1 & 0 & 0 & 0 \\ \frac{\alpha u_2}{\beta\rho M_{\text{art}}} & 0 & 1 & 0 & 0 \\ \frac{\alpha u_3}{\beta\rho M_{\text{art}}} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Here, the four free parameters

$$\delta \geq 0, \quad \beta > 0, \quad \alpha \geq 0 \quad \text{and} \quad M_{\text{art}} > 0$$

need to be determined. The effect of the modification (3.49) depends on this choice. Unfortunately, to the author's knowledge no strict parameter study has been performed to investigate the influence of such a choice. In our implementation we have chosen

$$\begin{aligned} \beta &:= a^2, \qquad \alpha = 0, && (3.52)\\ M_{\text{art}} &:= \min\left\{\max\left\{M^2, \kappa M_\infty^2\right\}, 1\right\} \\ a_{\text{art}} &:= \sqrt{\tau^2 \frac{V^2}{A^2} + M_{\text{art}} a^2}, \qquad \tau := \frac{1}{2}\left(1 - M_{\text{art}}\right). && (3.53) \end{aligned}$$

The modification M_{art} is called artificial Mach number and a_{art} describes a corresponding artificial speed of sound. The additional parameter κ was chosen by $\kappa = 1$ for all the computations shown in this paper. Often in computer codes this is a user defined parameter. The further parameter δ may be either chosen by

$$\delta := \begin{cases} 0, & M^2 \geq 1, \\ 1, & \text{else}, \end{cases} \qquad \text{or} \tag{3.54a}$$

$$\delta := \begin{cases} 0, & M^2 \geq 1, \\ 1 - M^n, n \geq 2 & \text{else}, \end{cases} \qquad \text{or} \tag{3.54b}$$

$$\delta := 0. \tag{3.54c}$$

Note that all criteria are designed such that the modification of the operator weighting the artificial viscosity are turned off when $|M| \geq 1$. This is a desirable effect as for transonic and supersonic flows no modification is required. Again, throughout this paper we used in our computations the simple choice (3.54c) for the definition of δ. The parameter choice (3.52) and (3.54c) considered are in agreement with [103, 111, 22].

3.3.3 Eigendecomposition of Turkel's modification

To implement a matrix valued artificial viscosity (3.49) the Eigendecomposition of $\mathbf{P}_{ij}^{-1}\left|\mathbf{P}_{ij}\mathbf{A}_{ij}^{\text{Roe}}\right|$ is required. To this end we start with the eigendecomposition of the modified system in entropy variables (3.50),

$$\mathbf{P}_{\mathbf{LM}}^{(0)} \frac{\partial\left\langle f_c\left(W_{prim}^{(0)}\right), n\right\rangle}{\partial W_{prim}^{(0)}} = \begin{pmatrix} M_{\text{art}}V & M_{\text{art}}n_1\rho a^2 & M_{\text{art}}n_2\rho a^2 & M_{\text{art}}n_3\rho a^2 & -M_{\text{art}}\delta V \\ \frac{n_1}{\rho} & V & 0 & 0 & 0 \\ \frac{n_2}{\rho} & 0 & V & 0 & 0 \\ \frac{n_3}{\rho} & 0 & 0 & V & 0 \\ 0 & 0 & 0 & 0 & V \end{pmatrix}.$$

Theorem 3.3.1 *The eigenvalues of* $\mathbf{P}_{\mathbf{LM}}^{(0)} \frac{\partial\left\langle f_c\left(W_{prim}^{(0)}\right), n\right\rangle}{\partial W_{prim}^{(0)}}$ *are given by*

$$\lambda_1 = \lambda_2 = \lambda_3 = V, \qquad \lambda_{4/5} = \frac{1}{2}\left(V + M_{\text{art}}V \pm T\right)$$

and the corresponding matrix of eigenvectors is given by

$$G^{(0)}_{prim,\mathbf{LM}} := \begin{pmatrix} 0 & 0 & 0 & \frac{a}{2}t_+ & -\frac{a}{2}t_- \\ -\delta\frac{V}{\rho} & -n_3\frac{a}{\rho} & n_2\frac{a}{\rho} & n_1\frac{a}{\rho} & -n_1\frac{a}{\rho} \\ n_3\frac{a}{\rho} & -\delta\frac{V}{\rho} & -n_1\frac{a}{\rho} & n_2\frac{a}{\rho} & -n_2\frac{a}{\rho} \\ -n_2\frac{a}{\rho} & n_1\frac{a}{\rho} & -\delta\frac{V}{\rho} & n_3\frac{a}{\rho} & -n_3\frac{a}{\rho} \\ -n_1a^2 & -n_2a^2 & -n_3a^2 & 0 & 0 \end{pmatrix}.$$

where $T := \sqrt{(M_{\text{art}}V - V)^2 + 4M_{\text{art}}a^2}$. *The inverse of the matrix of eigenvectors is given by*

$$\left(G^{(0)}_{prim,\mathbf{LM}}\right)^{-1} := \begin{pmatrix} 0 & -\delta V\frac{n^{(2)}_{23}\rho}{q_\delta} & \frac{\rho(n_3a+n_{12}\delta V)}{q_\delta} & -\frac{\rho(n_2a-n_{13}\delta V)}{q_\delta} & \frac{n_1}{a^2} \\ 0 & -\frac{\rho(n_3a-n_{12}\delta V)}{q_\delta} & -\delta V\frac{n^{(2)}_{31}\rho}{q_\delta} & \frac{\rho(n_1a+n_{23}\delta V)}{q_\delta} & \frac{n_2}{a^2} \\ 0 & \frac{\rho(n_2a+n_{13}\delta V)}{q_\delta} & -\frac{\rho(n_1a-n_{23}\delta V)}{q_\delta} & -\delta V\frac{n^{(2)}_{12}\rho}{q_\delta} & \frac{n_3}{a^2} \\ \frac{1}{aT} & -\frac{1}{2}\rho n_1\frac{t_-}{aT} & -\frac{1}{2}\rho n_2\frac{t_-}{aT} & -\frac{1}{2}\rho n_3\frac{t_-}{aT} & \frac{1}{2}V\delta\frac{t_-}{a^3T} \\ \frac{1}{aT} & -\frac{1}{2}\rho n_1\frac{t_+}{aT} & -\frac{1}{2}\rho n_2\frac{t_+}{aT} & -\frac{1}{2}\rho n_3\frac{t_+}{aT} & \frac{1}{2}V\delta\frac{t_+}{a^3T} \end{pmatrix},$$

where

$$\begin{aligned} n_{12} &:= n_1n_2, \qquad n_{13} := n_1n_3, \qquad n_{23} := n_2n_3 \\ n^{(2)}_{23} &:= n_2^2 + n_3^2, \qquad n^{(2)}_{31} := n_3^2 + n_1^2, \qquad n^{(2)}_{12} := n_1^2 + n_2^2, \\ q_\delta &:= \delta^2V^2 + a^2, \\ t_+ &:= M_{\text{art}}V - V + T, \qquad t_- := M_{\text{art}}V - V - T. \end{aligned}$$

Proof: The theorem follows by straightforward computations.

□

Since

$$\begin{aligned} &\mathbf{P}_{\mathbf{LM}}\frac{\partial\left\langle f_c\left(W_{\text{Roe}}\right), n\right\rangle}{\partial W} = \frac{\partial W}{\partial W^{(0)}_{prim}}\mathbf{P}^{(0)}_{\mathbf{LM}}\frac{\partial\left\langle f_c\left(W_{\text{Roe}}\right), n\right\rangle}{\partial W^{(0)}_{prim}}\frac{\partial W^{(0)}_{prim}}{\partial W} \\ &= \frac{\partial W}{\partial W^{(0)}_{prim}}G^{(0)}_{prim,\mathbf{LM}}\text{diag}(\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5)\left(G^{(0)}_{prim,\mathbf{LM}}\right)^{-1}\frac{\partial W^{(0)}_{prim}}{\partial W} \end{aligned}$$

we conclude

$$\left(G^{(0)}_{prim,\mathbf{LM}}\right)^{-1}\frac{\partial W^{(0)}_{prim}}{\partial W}\mathbf{P}_{\mathbf{LM}}\frac{\partial\left\langle f_c, n\right\rangle}{\partial W}\frac{\partial W}{\partial W^{(0)}_{prim}}G^{(0)}_{prim,\mathbf{LM}} = \text{diag}(\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5).$$

Therefore, the matrix of eigenvectors of the low Mach number preconditioned convective flux derivative $\mathbf{P}_{\mathbf{LM}}\frac{\partial\langle f_c(W_{\text{Roe}}),n\rangle}{\partial W}$ and its inverse with respect to conservative

variables are given by

$$\begin{aligned} G_{cons,\mathbf{LM}} &= \frac{\partial W}{\partial W^{(0)}_{prim}} G^{(0)}_{prim,\mathbf{LM}}, \\ (G_{cons,\mathbf{LM}})^{-1} &= \left(G^{(0)}_{prim,\mathbf{LM}}\right)^{-1} \frac{\partial W^{(0)}_{prim}}{\partial W}. \end{aligned}$$

Finally, a representation of the inverse of the low Mach number preconditioner $\left[\mathbf{P}^{(0)}_{\mathbf{LM}}\right]^{-1}$ in conservative variables is required. It is given by

$$[\mathbf{P}_{\mathbf{LM}}]^{-1} = \frac{\partial W}{\partial W^{(0)}_{prim}} \left[\mathbf{P}^{(0)}_{\mathbf{LM}}\right]^{-1} \frac{\partial W^{(0)}_{prim}}{\partial W} =$$

$$\begin{pmatrix} \frac{\beta_2}{M_{\text{art}}} + (\delta-1)(\beta_2-1) & -u_1\beta_6 & -u_2\beta_6 & -u_3\beta_6 & \beta_6 \\ u_1(\beta_1\beta_2-\delta) & -u_1^2\beta_6+1 & -u_1u_2\beta_6 & -u_1u_3\beta_6 & u_1\beta_6 \\ u_2(\beta_1\beta_2-\delta) & -u_2u_1\beta_6 & -u_2^2\beta_6+1 & -u_2u_3\beta_6 & u_2\beta_6 \\ u_3(\beta_1\beta_2-\delta) & -u_3u_1\beta_6 & -u_3u_2\beta_6 & -u_3^2\beta_6+1 & u_3\beta_6 \\ \beta_3\beta_4 - \|u\|_2^2 - \frac{(\beta_5-a^2)M^2}{2} & u_1\beta_7 & u_2\beta_7 & u_3\beta_7 & \frac{1}{M_{\text{art}}}+\delta+\beta_1\beta_2 \end{pmatrix},$$

where

$$\begin{aligned} \beta_1 &:= \delta + \frac{1}{M_{\text{art}}} - 1, \qquad \beta_2 := \frac{\gamma-1}{2}M^2 \\ \beta_3 &:= \frac{1}{\gamma-1} + \frac{M^2}{2}, \qquad \beta_4 := \frac{(\gamma-1)\|u\|_2^2}{2M_{\text{art}}} + \delta\left(\frac{(\gamma-1)\|u\|_2^2}{2} - a^2\right) \\ \beta_5 &:= \frac{(\gamma-1)\|u\|_2^2}{2}, \quad \beta_6 := \frac{(\gamma-1)\beta_1}{a^2}, \quad \beta_7 := (\beta_2-1)\beta_1. \end{aligned}$$

Finally, to avoid zero dissipation a classical entropy fix is integrated in the construction of the operator. To this end we define the operator

$$\begin{aligned} |\Lambda|_{\text{ef},\mathbf{LM}} &:= \operatorname{diag}\left(|\lambda|_{\text{ef},\mathbf{LM},1}, |\lambda|_{\text{ef},\mathbf{LM},2}, |\lambda|_{\text{ef},\mathbf{LM},3}, |\lambda|_{\text{ef},\mathbf{LM},4}, |\lambda|_{\text{ef},\mathbf{LM},5}\right), \\ |\lambda|_{\text{ef},\mathbf{LM},\text{i}} &:= \max\left\{|V|, \delta_{\text{ef}}\lambda_{\max,\mathbf{LM}}\right\}, \qquad i = 1,2,3, \\ |\lambda|_{\text{ef},\mathbf{LM},4} &:= \max\left\{|V + M_{\text{art}}V + T|, \delta_{\text{ef}}\lambda_{\max,\mathbf{LM}}\right\}, \\ |\lambda|_{\text{ef},\mathbf{LM},5} &:= \max\left\{|V + M_{\text{art}}V - T|, \delta_{\text{ef}}\lambda_{\max,\mathbf{LM}}\right\}, \\ \lambda_{\max,\mathbf{LM}} &:= \max\left\{|V|, \frac{1}{2}|V + M_{\text{art}}V + T|, \frac{1}{2}|V + M_{\text{art}}V - T|\right\}. \end{aligned}$$

Then the operator $\mathbf{P}_{ij}^{-1}\left|\mathbf{P}_{ij}\mathbf{A}_{ij}^{\text{Roe}}\right|$ required for the implementation in (3.49) is given via

$$\mathbf{P}_{ij}^{-1}\left|\mathbf{P}_{ij}\mathbf{A}_{ij}^{\text{Roe}}\right| = [\mathbf{P}_{\mathbf{LM}}]^{-1} G_{cons,\mathbf{LM}} |\Lambda|_{\text{ef},\mathbf{LM}} (G_{cons,\mathbf{LM}})^{-1}. \tag{3.55}$$

To evaluate the operators on the right hand side of (3.55) Roe-averaged variables are used on the edge. In all our computations we chose $\delta_{\text{ef}} = 0.2$.

3.3.4 Extension of Roe matrix to the incompressible limit

Contrary to the modification of the artificial viscosity in (3.49), where the full weighting operator was changed by some matrix $\mathbf{P}_{ij}$, it is possible to alter only some terms of $\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|_{\mathrm{ef}}$ in (3.16) to extend the numerical flux function (3.15) to the incompressible limit. This idea goes back to Rossow [78, 79] and was successfully applied by Rossow, Turkel and Swanson [93, 91]. Using the representation (3.39) of $\left|\frac{\partial\langle f_c(W_{\mathrm{Roe}}),n\rangle}{\partial W}\right|_{\mathrm{ef}}$, the speed of sound a in (3.39) is replaced by the artificial speed of sound (3.53).

In opposite to the observation of Rossow, this change in the construction of the operator (3.39) was not successful at all. It turned out that a further modification was necessary.

Additionally, in our implementation we modify the absolute value of the normal velocity by

$$|V|_{\mathrm{ef},1,\mathrm{art}} := \max\left\{|V|, \omega_{\mathrm{ef}}\delta_{\mathrm{ef}}\left(|V| + a_{\mathrm{art}}A\right)\right\}. \tag{3.56}$$

Here ω_{ef} is a further parameter, and it was chosen by $\omega_{\mathrm{ef}} = 3$ in all our computations. And, besides the replacement of the speed of sound a by the artificial speed of sound a_{art} in all formulae (3.39a)– (3.39i), as a second ingredient, we only replace in (3.39e) the absolute value of the normal velocity $|V|_{\mathrm{ef},1}$ by the modified one, $|V|_{\mathrm{ef},1,\mathrm{art}}$.

3.4 Discretization: The viscous part

To discretize the viscous part f_v of (2.45) we notice that due to the definitions of the viscous stress tensor τ in (2.7) and because of (2.8) derivatives of the velocity u and the temperature T are required. Due to (3.11) the ansatz function (3.9) does not support the representation of gradients. Hence, other ways need to be established to incorporate such information into the discretization scheme. In the following Section we give two different possibilities to define a discrete approximation to the derivative.

3.4.1 Approximation of gradients

Our approach to discretize the viscous terms f_v in (2.45) is to define the required derivatives on the corresponding faces such that the surface integrals can be directly evaluated.

Definition 3.4.1 *Assume that M is a triangulation of D and $D_i, D_j \in M$, $i \neq j$. Consider the function*

$$g(x) := \sum_{i=1}^{N_{elem}} g_i \mathbb{1}_{D_i}(x), \qquad g_i \in \mathbb{R}. \tag{3.57}$$

Then we define its mean value on the face e_{ij} by

$$\mathbb{1}_{D_{e_{ij}}}(y) := \frac{1}{2}\left(\lim_{h\to+0}\mathbb{1}_{D_i}(y+hn_{e_{ij}}) + \lim_{h\to+0}\mathbb{1}_{D_j}(y-hn_{e_{ij}})\right), \qquad y\in\partial D_i\cap\partial D_j,$$

where $\lim_{h\to+0}\mathbb{1}_{D_\ell}(y+hn_{e_{ij}})$ denotes the continuous continuation to the boundary of D_ℓ, $\ell=i,j$.

Definition 3.4.2 *Assume that M is a triangulation of D and $D_\ell\in M$. Let us consider the function g given in (3.57). Then we call for $k=1,2,3$,*

$$\left(\frac{\partial g(x)}{\partial x_k}\right)_{D_\ell}^{\mathrm{GG}} := \frac{\mathbb{1}_{D_\ell}(x)}{\mathrm{vol}\,(D_\ell)}\int_{\partial D_\ell}\langle g(y), P_k n\rangle\, ds(y) \tag{3.58}$$

the Green-Gauss method to approximate the derivative in direction x_k in the control volume D_ℓ. Here P_k denotes the orthogonal projection

$$P_k x = \langle x, e_k\rangle\, e_k. \tag{3.59}$$

The implementation of the Green-Gauss method is straightforward due to the following observation:

Lemma 3.4.3 *Assume that M is a triangulation of D and $D_\ell\in M$. Let us consider the function (3.57). Then the Green-Gauss method to approximate the derivative in direction x_k for the control volume D_ℓ is given by*

$$\left(\frac{\partial g(x)}{\partial x_k}\right)_{D_\ell}^{\mathrm{GG}} = \begin{cases} \frac{1}{\mathrm{vol}(D_\ell)}\sum_{j\in\mathcal{N}(\ell)}\mathrm{svol}(e_{\ell j})\frac{n_{k,\ell j}}{2}\left(g_\ell+g_j\right), & x\in D_\ell \\ 0, & else. \end{cases} \tag{3.60}$$

Proof: For $x\in D_\ell$ the definition (3.58) can be rewritten as

$$\begin{aligned}\left(\frac{\partial g(x)}{\partial x_k}\right)_{D_\ell}^{\mathrm{GG}} &= \frac{1}{\mathrm{vol}\,(D_\ell)}\sum_{j\in\mathcal{N}(\ell)}\mathrm{svol}(e_{\ell j})\frac{n_{k,\ell j}}{2}\left(g_\ell\lim_{h\to+0}\mathbb{1}_{\overline{D}_\ell}(x+hn_{e_{ij}})\right.\\ &\qquad\left.+\lim_{h\to+0}g_j\mathbb{1}_{\overline{D}_j}(x-hn_{e_{ij}})\right)\\ &= \frac{1}{\mathrm{vol}\,(D_\ell)}\sum_{j\in\mathcal{N}(\ell)}\mathrm{svol}(e_{\ell j})\frac{n_{k,\ell j}}{2}\left(g_\ell+g_j\right).\end{aligned}$$

This proves the assertion.

□

Lemma 3.4.4 *Assume that M is a triangulation of D and $D_\ell \in M$ such that D_ℓ and its six direct neighbor cells D_j, $j \in \mathcal{N}(\ell)$, are rectangular and of the same size, that is*

$$\begin{aligned}\overline{D}_\ell = \{x \in \mathbb{R}^m \ : \ & x = a_\ell + x_1 e_1 + x_2 e_2 + x_3 e_3, \\ & 0 \le x_1 \le h_1, 0 \le x_2 \le h_2, 0 \le x_3 \le h_3\}.\end{aligned}$$

$$\begin{aligned}\overline{D}_j = \{x \in \mathbb{R}^m \ : \ & x = a_j + x_1 e_1 + x_2 e_2 + x_3 e_3, \\ & 0 \le x_1 \le h_1, 0 \le x_2 \le h_2, 0 \le x_3 \le h_3\}.\end{aligned}$$

where

$$a_j = a_\ell + h_j, \qquad a_{j+3} = a_\ell - h_j, \qquad j = 1, 2, 3.$$

Let us consider the linear function

$$f(x) = \langle b, x \rangle, \qquad a \in \mathbb{R}^m, \tag{3.61}$$

which is approximated by the function (3.57) with $g_i = f(p_i)$, $i = 1, \ldots, N_{elem}$. Then the Green-Gauss method to approximate the derivative of f is exact, that is

$$\frac{\partial f(x)}{\partial x_k} = \left(\frac{\partial g(x)}{\partial x_k}\right)^{\mathrm{GG}}_{D_\ell}.$$

Proof: We only prove the assertion for $\frac{\partial f(x)}{\partial x_1} = b_1$. The other directions are analogously. First of all note that

$$\begin{aligned}
\mathrm{vol}(D_\ell) &= h_1 h_2 h_3, && (3.62a)\\
\mathrm{svol}(e_{\ell j_1}) &= h_2 h_3, \quad \mathrm{svol}(e_{\ell j_2}) = h_1 h_3, \quad \mathrm{svol}(e_{\ell j_3}) = h_1 h_2, && (3.62b)\\
\mathrm{svol}(e_{\ell j_4}) &= h_2 h_3, \quad \mathrm{svol}(e_{\ell j_5}) = h_1 h_3, \quad \mathrm{svol}(e_{\ell j_6}) = h_1 h_2, && (3.62c)\\
n_{1,\ell j_1} &= 1, \qquad n_{1,\ell j_2} = 0, \qquad n_{1,\ell j_3} = 0, && (3.62d)\\
n_{1,\ell j_4} &= -1, \qquad n_{1,\ell j_5} = 0, \qquad n_{1,\ell j_6} = 0, && (3.62e)\\
p_\ell &= a_\ell + \frac{1}{2}\left(\sum_{j=1}^{3} h_j e_j\right), && (3.62f)\\
p_{j_1} &= a_\ell + \frac{3}{2} h_1 e_1 + \frac{1}{2} h_2 e_2 + \frac{1}{2} h_3 e_3, && (3.62g)\\
p_{j_4} &= a_\ell - \frac{1}{2} h_1 e_1 + \frac{1}{2} h_2 e_2 + \frac{1}{2} h_3 e_3. && (3.62h)\\
& && (3.62i)
\end{aligned}$$

Then formula (3.60) gives

$$\begin{aligned}
\left(\frac{\partial g(x)}{\partial x_1}\right)^{\mathrm{GG}}_{D_\ell} &= \frac{1}{2h_1 h_2 h_3}\left\{[h_2 h_3 (f(p_\ell) + f(p_{j_1}))] - [h_2 h_3 (f(p_\ell) + f(p_{j_4}))]\right\} \\
&= \frac{1}{2h_1}\left\{[2\langle b, a_\ell\rangle + 2h_1 b_1 + h_2 b_2 + h_3 b_3] - [2\langle b, a_\ell\rangle + h_2 b_2 + h_3 b_3]\right\} \\
&= b_1,
\end{aligned}$$

which proves the assertion.

□

Formula (3.58) gives a method to approximate the gradient of a function at the barycenter of a control volume. To discretize the viscous terms of (2.1a) we follow the idea to directly evaluate the terms on a face. To this end derivatives on a face are required.

Definition 3.4.5 *Assume that M is a triangulation of D and $e_{ij} \in E(M)$. Let us consider the function (3.57). Then the Green-Gauss method to approximate the derivative in direction x_k on the face e_{ij} is given by*

$$\left[\frac{\partial g(x)}{\partial x_k}\right]_{e_{ij}}^{\mathrm{GG}} := \frac{1}{2}\left[\left(\frac{\partial g(x)}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} + \left(\frac{\partial g(x)}{\partial x_k}\right)_{D_j}^{\mathrm{GG}}\right]. \tag{3.63}$$

Corollary 3.4.6 *Assume that M is a Cartesian triangulation of D. Then the Green-Gauss method to approximate the derivative of the linear function f given by (3.61) is exact for all $x \in D$. In particular, the Green-Gauss method to approximate the derivative in direction x_k on each face $e_{ij} \in E(M)$ is exact.*

Proof: This is a direct consequence of Lemma 3.4.4 and the definition (3.63).

□

A second method to directly approximate the derivative of a function on the face e_{ij} is given by the so called thin layer approximation.

Definition 3.4.7 *Assume that M is a triangulation of D and $e_{ij} \in E(M)$. Let us consider the function (3.57). Then the thin shear layer* (TSL) *method to approximate the derivative in direction x_k on the face e_{ij} is given by*

$$\left(\frac{\partial g(x)}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} := \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\begin{cases} \left(g_j \mathbb{1}_{D_j}(p_j) - g_i \mathbb{1}_{D_i}(p_i)\right), & x \in e_{ij}, \\ 0, & else. \end{cases} \tag{3.64}$$

Formula (3.64) gives a direct possibility to approximate the derivative on the face and for $x \in e_{ij}$ it can be shorter written by

$$\left(\frac{\partial g(x)}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = n_{k,ij}\frac{(g_j - g_i)}{\mathrm{dist}(e_{ij})}. \tag{3.65}$$

Lemma 3.4.8 *Consider the assumption of Lemma 3.4.4. Then the thin shear layer method to approximate the derivative of f on the face $e_{\ell j} \in E(M)$ is exact in normal direction, that is it is exact for*

$$\tilde{f}_k(x) = \langle b, P_k x\rangle$$

and P_k is given in (3.59).

Proof: Recall the relations (3.62). Then, using $\mathrm{dist}(e_{\ell j}) = h_1$, we compute

$$\begin{aligned}\left(\frac{\partial g(x)}{\partial x_1}\right)_{e_{\ell j}}^{\mathrm{TSL}} &= \frac{1}{h_1}\left[\left(\langle b, a_\ell\rangle + \frac{3}{2}h_1 b_1 + \frac{1}{2}h_2 b_2 + \frac{1}{2}h_3 b_3\right)\right.\\ &\quad \left. - \left(\langle b, a_\ell\rangle + \frac{1}{2}h_1 b_1 + \frac{1}{2}h_2 b_2 + \frac{1}{2}h_3 b_3\right)\right]\\ &= b_1.\end{aligned}$$

Since $n_{2,\ell j} = n_{3,\ell j} = 0$ we obtain $\left(\frac{\partial g(x)}{\partial x_2}\right)_{e_{\ell j}}^{\mathrm{TSL}} = \left(\frac{\partial g(x)}{\partial x_3}\right)_{e_{\ell j}}^{\mathrm{TSL}} = 0$, which shows that the thin layer method is exact for $\tilde{f}_1$. The assertion for $\tilde{f}_2$ and $\tilde{f}_3$ is analogously. □

3.4.2 Discretization of viscous flux terms

Using (3.58) together with (3.63) and (3.64) give us two different ways to approximate derivatives to implement the viscous terms. Both methods are explicit and straightforward to implement. On the other hand, both methods have severe shortcomings.

For the Green-Gauss method it can only be shown that for Cartesian meshes the derivative of linear functions can be correctly represented. The thin shear layer method even makes a systematic error. Only the derivative of the linear function projected on the normal direction can be correctly represented on a Cartesian mesh.

Both approximate derivatives (3.58) and (3.64) are a severe source of error in the typical finite volume discretization of the system of equations (2.45). This is in particular true for unstructured meshes where many of the control volumes are non hexahedral elements. On the other hand, many of the grids used in practice have a hexahedral boundary layer. Assuming that in particular in the boundary layer derivatives of certain quantities such as velocity u have a significant impact, at least the topology of the mesh is adapted to the shortcomings of the approximate derivatives (3.58) and (3.64). Nevertheless, it should be pointed out that even under optimal conditions, that is a Cartesian mesh, only the derivative of a linear function can be represented correctly.

To discretize the viscous terms f_v of (2.45) an averaging of all required data on the face e_{ij} is done,

$$\mu_{l,e_{ij}} = \frac{1}{2}\left(\mu_l(W_i) + \mu_l(W_j)\right), \qquad \mu_{t,e_{ij}} = \frac{1}{2}\left(\mu_{t,i} + \mu_{t,j}\right), \tag{3.66}$$

$$\kappa_{l,e_{ij}} = \frac{1}{2}\left(\kappa_l(W_i) + \kappa_l(W_j)\right), \qquad \kappa_{t,e_{ij}} = \frac{1}{2}\left(\kappa_{t,i} + \kappa_{t,j}\right), \tag{3.67}$$

$$\mu_{\mathrm{eff},e_{ij}} = \mu_{l,e_{ij}} + \mu_{t,e_{ij}}, \qquad \kappa_{\mathrm{eff},e_{ij}} = \kappa_{l,e_{ij}} + \kappa_{t,e_{ij}},$$

$$u_{e_{ij}} = \frac{1}{2}\left(u_i + u_j\right). \tag{3.68}$$

In case we use the Green-Gauss method the required gradients are computed by (3.60) and (3.63), in case we use the thin layer method we use (3.65). Then, for completeness, the required gradients for velocity and temperature are either

$$\left(\frac{\partial u}{\partial x_k}\right)_{e_{ij}}^{\mathrm{GG}} = \frac{1}{2}\left[\left(\frac{\partial u}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} + \left(\frac{\partial u}{\partial x_k}\right)_{D_j}^{\mathrm{GG}}\right], \tag{3.69a}$$

$$\left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{GG}} = \frac{1}{2}\left[\left(\frac{\partial T}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} + \left(\frac{\partial T}{\partial x_k}\right)_{D_j}^{\mathrm{GG}}\right] \tag{3.69b}$$

or

$$\left(\frac{\partial u}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\left(u_j - u_i\right). \tag{3.70a}$$

$$\left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\left(T_j - T_i\right). \tag{3.70b}$$

The terms (3.66), (3.67), (3.68), and (3.69) or (3.70) are used to evaluate the viscous stress tensor τ and θ defined in (2.7) and (2.8). The corresponding flux for the face e_{ij} is then evaluated by

$$\int_{e_{ij}} \left\langle f_v\left(W_h\right), n_{e_{ij}}\right\rangle ds \approx \mathrm{svol}\left(e_{ij}\right)\left\langle f_v\left(W_{\mathcal{N}(i,j)}\right), n_{e_{ij}}\right\rangle. \tag{3.71}$$

Let us close this section with one important remark. In case the gradients are evaluated using (3.69), the viscous flux (3.71) depends on the ansatz functions $W_{\mathcal{N}(i,j)}$ defined in (3.14), which means an extended stencil. In case (3.70) is used, the stencil for the approximation of the viscous flux (3.71) is compact. It only depends on W_i and W_j.

3.5 Discretization of turbulence models

The way to discretize the turbulent flow equations (2.52) and (2.53) follow the same ideas compared to those for the mean flow equations. Additionally, a discretization strategy of the integral operator $V_D\left(Q\right)$ representing the source terms is required. A major difference in the discretization of turbulent flow equations compared to the mean flow equations is the restriction of the convection parts to first order terms.

According to (3.9) we introduce the ansatz functions to approximate the equations

for $\tilde{\nu}$, k and ω,

$$\begin{aligned}
\tilde{\nu}(x,t) &\approx \tilde{\nu}_h(x,t), \qquad \tilde{\nu}_h(x,t) := \sum_{i=1}^{N_{elem}} \tilde{\nu}_i(t)\mathbb{1}_{D_i}(x), \qquad \tilde{\nu}_i(t) := \tilde{\nu}\left(p_i,t\right), \\
k(x,t) &\approx k_h(x,t), \qquad k_h(x,t) := \sum_{i=1}^{N_{elem}} k_i(t)\mathbb{1}_{D_i}(x), \qquad k_i(t) := k\left(p_i,t\right), \\
\omega(x,t) &\approx \omega_h(x,t), \qquad \omega_h(x,t) := \sum_{i=1}^{N_{elem}} \omega_i(t)\mathbb{1}_{D_i}(x), \qquad \omega_i(t) := \omega\left(p_i,t\right).
\end{aligned}$$

Definition 3.5.1 *Assume that M is a triangulation of D, $e_{ij} \in E(M)$ and let us consider the corresponding control volumes $D_i, D_j \in M$. Then we define the numerical flux function for the Spalart-Allmaras model in normal direction n by*

$$\begin{aligned}
\mathcal{H}_{\mathrm{SA}}\left(\left(\tilde{\nu}_i, W_i\right), \left(\tilde{\nu}_j, W_j\right), n\right) &:= \frac{1}{2}\left[\left\langle f_{c,\mathrm{SA}}\left(\tilde{\nu}_i, W_i\right), n\right\rangle + \left\langle f_{c,\mathrm{SA}}\left(\tilde{\nu}_j, W_j\right), n\right\rangle\right] \\
&\quad -\frac{1}{2}|V_{ij,\mathrm{Roe}}|\left(\tilde{\nu}_j - \tilde{\nu}_i\right).
\end{aligned} \tag{3.72}$$

The term $V_{ij,\mathrm{Roe}}$ is declared in Definition 3.2.3.

Definition 3.5.2 *Assume that M is a triangulation of D, $e_{ij} \in E(M)$ and let us consider the corresponding control volumes $D_i, D_j \in M$. Then we define the numerical flux function for the $k\omega$-model in normal direction n by*

$$\mathcal{H}_{(k,\omega)}\left(\left(k_i, \omega_i, W_i\right), \left(k_j, \omega_j, W_j\right), n\right) := \begin{cases} \begin{cases} \left\langle u_{e_{ij}}, n\right\rangle k_i, & \left\langle u_{e_{ij}}, n\right\rangle \geq 0, \\ \left\langle u_{e_{ij}}, n\right\rangle k_j, & \left\langle u_{e_{ij}}, n\right\rangle < 0, \end{cases} \\ \begin{cases} \left\langle u_{e_{ij}}, n\right\rangle \omega_i, & \left\langle u_{e_{ij}}, n\right\rangle \geq 0, \\ \left\langle u_{e_{ij}}, n\right\rangle \omega_j, & \left\langle u_{e_{ij}}, n\right\rangle < 0. \end{cases} \end{cases} \tag{3.73}$$

It can be shown for both $\mathcal{H}_{\mathrm{SA}}$ and $\mathcal{H}_{(k,\omega)}$ that these satisfy the conditions of Definition 3.2.3. And both flux functions are formulated and implemented without an entropy fix. The convective flux for the Spalart-Allmaras turbulence model is approximated by

$$\int_{e_{ij}} \left\langle f_{c,\mathrm{SA}}\left(W_h\right), n_{e_{ij}}\right\rangle ds \approx \mathrm{svol}\left(e_{ij}\right) \mathcal{H}_{\mathrm{SA}}\left(\left(\tilde{\nu}_i, W_i\right), \left(\tilde{\nu}_j, W_j\right), n_{e_{ij}}\right),$$

and for the $k\omega$-model by

$$\int_{e_{ij}} \left\langle f_{c,(k,\omega)}\left(W_h\right), n_{e_{ij}}\right\rangle ds \approx \mathrm{svol}\left(e_{ij}\right) \mathcal{H}_{(k,\omega)}\left(\left(k_i, \omega_i, W_i\right), \left(k_j, \omega_j, W_j\right), n_{e_{ij}}\right).$$

The discretization of the viscous flux terms follows the ideas of Section 3.4.2. Additional gradients for $\tilde{\nu}$, k and ω are computed either by

$$\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)^{\mathrm{GG}}_{e_{ij}} = \frac{1}{2}\left[\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)^{\mathrm{GG}}_{D_i} + \left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)^{\mathrm{GG}}_{D_j}\right], \tag{3.74a}$$

$$\left(\frac{\partial k}{\partial x_k}\right)^{\mathrm{GG}}_{e_{ij}} = \frac{1}{2}\left[\left(\frac{\partial k}{\partial x_k}\right)^{\mathrm{GG}}_{D_i} + \left(\frac{\partial k}{\partial x_k}\right)^{\mathrm{GG}}_{D_j}\right], \tag{3.74b}$$

$$\left(\frac{\partial\omega}{\partial x_k}\right)^{\mathrm{GG}}_{e_{ij}} = \frac{1}{2}\left[\left(\frac{\partial\omega}{\partial x_k}\right)^{\mathrm{GG}}_{D_i} + \left(\frac{\partial\omega}{\partial x_k}\right)^{\mathrm{GG}}_{D_j}\right] \tag{3.74c}$$

or

$$\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)^{\mathrm{TSL}}_{e_{ij}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\left(\tilde{\nu}_j - \tilde{\nu}_i\right), \tag{3.75a}$$

$$\left(\frac{\partial k}{\partial x_k}\right)^{\mathrm{TSL}}_{e_{ij}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\left(k_j - k_i\right), \tag{3.75b}$$

$$\left(\frac{\partial\omega}{\partial x_k}\right)^{\mathrm{TSL}}_{e_{ij}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\left(\omega_j - \omega_i\right). \tag{3.75c}$$

Simple averaging of

$$\nu_{l,e_{ij}} = \frac{1}{2}\left(\nu_l\left(W_i\right) + \nu_l\left(W_j\right)\right), \tag{3.76}$$

$$\tilde{\nu}_{e_{ij}} = \frac{1}{2}\left(\tilde{\nu}_i + \tilde{\nu}_j\right), \quad f_{n,e_{ij}} = \frac{c_{n1} + \frac{1}{2}\left(\chi\left(\tilde{\nu}_i, W_i\right)^3 + \chi\left(\tilde{\nu}_j, W_j\right)^3\right)}{c_{n1} - \frac{1}{2}\left(\chi\left(\tilde{\nu}_i, W_i\right)^3 + \chi\left(\tilde{\nu}_j, W_j\right)^3\right)}, \tag{3.77}$$

$$\left(\frac{k}{\omega}\right)_{e_{ij}} = \frac{1}{2}\left(\frac{k_i}{\omega_i} + \frac{k_j}{\omega_j}\right), \qquad \Gamma(T)_{e_{ij}} = \Gamma(T_i) + \Gamma(T_j), \tag{3.78}$$

gives all required values to evaluate the viscous terms of the SA-model

$$\int_{e_{ij}} \left\langle f_{v,\mathrm{SA}}\left(\tilde{\nu}_h, W_h\right), n_{e_{ij}}\right\rangle ds \approx \mathrm{svol}\left(e_{ij}\right)\left\langle f_{v,\mathrm{SA}}\left(\tilde{\nu}_{\mathcal{N}(i,j)}, W_{\mathcal{N}(i,j)}\right), n_{e_{ij}}\right\rangle,$$

and those of the $k\omega$-model,

$$\begin{aligned}
&\begin{pmatrix} \omega_{sc}^{-1} & 0 \\ 0 & \omega_{sc}^{-1} \end{pmatrix} \int_{e_{ij}} \left\langle \tilde{f}_{v,(k,\omega)}\left(k_h, \omega_h, W_h\right), n_{e_{ij}}\right\rangle ds \\
\approx\ &\begin{pmatrix} \omega_{sc}^{-1} & 0 \\ 0 & \omega_{sc}^{-1} \end{pmatrix} \mathrm{svol}\left(e_{ij}\right) \left\langle \tilde{f}_{v,(k,\omega)}\left(k_{\mathcal{N}(i,j)}, \omega_{\mathcal{N}(i,j)}, W_{\mathcal{N}(i,j)}\right), n_{e_{ij}}\right\rangle.
\end{aligned}$$

Additionally, for the turbulence flow equations the source terms Q_{SA} and $Q_{k,\omega}$ need to be discretized. This is done pointwise by the straightforward application of the mid-point rule

$$\int_{D_i} Pr_{\mathrm{SA}}\left(\tilde{\nu}_h, W_h\right) dx \approx \mathrm{vol}\left(D_i\right) Pr_{\mathrm{SA}}\left(\tilde{\nu}_i, W_i, W_{j,j\in\mathcal{N}(i)}\right), \tag{3.79a}$$

$$\int_{D_i} De_{\mathrm{SA}}\left(\tilde{\nu}_h, W_h\right) dx \approx \mathrm{vol}\left(D_i\right) De_{\mathrm{SA}}\left(\tilde{\nu}_i, W_i, W_{j,j\in\mathcal{N}(i)}\right), \tag{3.79b}$$

$$\int_{D_i} Di_{\mathrm{SA}}\left(\tilde{\nu}_h, W_h\right) dx \approx \mathrm{vol}\left(D_i\right) Di_{\mathrm{SA}}\left(\tilde{\nu}_i, \tilde{\nu}_{j,j\in\mathcal{N}(i)}\right). \tag{3.79c}$$

The notation on the right hand sides of (3.79) is understood as follows. The construction of Pr_{SA} and De_{SA} requires the derivative of the velocity and Di_{SA} of the transported variable $\tilde{\nu}$. These are computed for the element D_i using (3.59). Hence, the formulation of the right hand sides of (3.79a) and (3.79b) indicate the dependency of the source term in the element D_i on the ansatz functions W_i and $W_{j,j\in\mathcal{N}(i)}$ and the dependency of the right hand side of (3.79c) on the ansatz functions $\tilde{\nu}_i$ and $\tilde{\nu}_{j,j\in\mathcal{N}(i)}$.

The discretization of the source terms for the $k\omega$-model is analogue,

$$\omega_{sc}^{-1}\int_{D_i} Pr_{k,(k,\omega)}\left(k_h, \omega_h, W_h\right) dx \approx \frac{\mathrm{vol}\left(D_i\right)}{\omega_{sc}} Pr_{k,(k,\omega)}\left(k_i, \omega_i, W_i, W_{j,j\in\mathcal{N}(i)}\right), \tag{3.80a}$$

$$\omega_{sc}\int_{D_i} De_{k,(k,\omega)}\left(k_h, \omega_h, W_h\right) dx \approx \omega_{sc}\mathrm{vol}\left(D_i\right) De_{k,(k,\omega)}\left(k_i, \omega_i\right), \tag{3.80b}$$

$$\omega_{sc}^{-1}\int_{D_i} Pr_{\omega,(k,\omega)}\left(k_h, \omega_h, W_h\right) dx \approx \frac{\mathrm{vol}\left(D_i\right)}{\omega_{sc}} Pr_{\omega,(k,\omega)}\left(W_i, W_{j,j\in\mathcal{N}(i)}\right), \tag{3.80c}$$

$$\omega_{sc}\int_{D_i} De_{\omega,(k,\omega)}\left(k_h, \omega_h, W_h\right) dx \approx \omega_{sc}\mathrm{vol}\left(D_i\right) De_{\omega,(k,\omega)}\left(\omega_i\right). \tag{3.80d}$$

And the right hand sides of (3.80) indicate as above the direct dependencies on the ansatz functions and coefficients. According to (2.23) the discretized production term $\tilde{Pr}_{k,(k,\omega)}$ is given by

$$\int_{D_i} \tilde{Pr}_{k,(k,\omega)} \approx \mathrm{vol}\left(D_i\right) \min\{\omega_{sc}^{-1} Pr_{k,(k,\omega)}, 20\omega_{sc} De_{k,(k,\omega)}\}. \tag{3.81}$$

3.6 Discretization of boundary conditions

All boundary conditions considered in this thesis are implemented using a flux formulation. The concept is explained at the beginning of this section once in its general formulation. Then details for the following boundary conditions are given:

a) Farfield boundary condition,

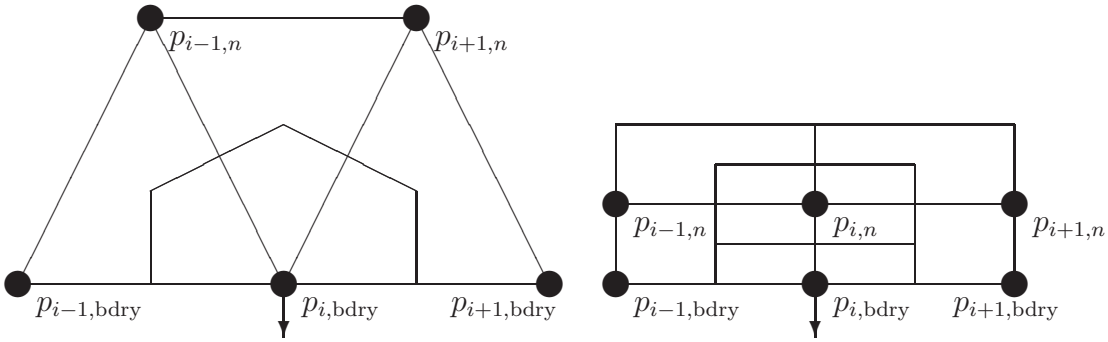

Figure 3.3: Examples of a polyhedral and quadrilateral mesh for a boundary point

b) Slip wall boundary condition,

c) No-slip wall boundary condition,

d) Symmetry boundary condition.

To explain the general idea consider Figure 3.3 and assume that $e_{i,\text{bdry}}$ represents a boundary edge. As already mentioned in the introduction of this section, there is an anomaly in the computational mesh considering a dual mesh. It can be observed from Figure 3.3 that due to construction the element at the boundary is given by a half cell, which directly results in a jump in the grid metric. Hence, the considered barycenter of this cell is assumed to be located directly on the boundary, that is as if this half cell is assumed to be artificially extended to a complete cell. Within our proposed formulation of the boundary conditions, we cannot fully circumvent this fact.

Using flux formulations, the flux over the edge $e_{i,\text{bdry}}$ needs to satisfy the corresponding boundary condition. Contrary to an inner edge, at the outset only one, namely the inner state is given. To be more exact, the coefficients W_i of the ansatz function corresponding to the element D_i are given. To compute the flux, an outer artificial state needs to be prescribed, which is denoted by,

$$W_{i,\text{bdry}} = \left(\rho_{i,\text{bdry}}(t), (\rho u)_{i,\text{bdry}}(t), (\rho E)_{i,\text{bdry}}(t)\right),$$

and additionally for the possible turbulence flow equations $\tilde{\nu}_{i,\text{bdry}}$ and $(k_{i,\text{bdry}}, \omega_{i,\text{bdry}})$. This outer artificial state is evaluated such that the boundary condition for the boundary edge $e_{i,\text{bdry}}$ is satisfied. Then, using the notation of Definition 3.1.12 we

approximate the boundary integrals by

$$\begin{aligned}
\int_{\partial D_i} \langle f_c(W), n\rangle \, ds &\approx \sum_{i=1}^{N_{\text{bdry}}} \int_{e_{i,\text{bdry}}} \mathcal{H}^{1st,\text{Roe}}(W_i, W_{i,\text{bdry}}, n_{i,\text{bdry}}) \, ds \\
&\approx \sum_{i=1}^{N_{\text{bdry}}} \text{svol}(e_{i,\text{bdry}}) \, \mathcal{H}^{1st,\text{Roe}}(W_i, W_{i,\text{bdry}}, n_{i,\text{bdry}}), \\
\int_{\partial D_i} \langle f_v(W), n\rangle \, ds &\approx \sum_{i=1}^{N_{\text{bdry}}} \text{svol}(e_{i,\text{bdry}}) \left\langle f_v\left(W_i, W_{j,j\in\mathcal{N}(i)}, W_{i,\text{bdry}}\right), n_{i,\text{bdry}}\right\rangle.
\end{aligned}$$

For stability reasons in particular for solid walls the numerical flux function for boundary edges is replaced by $\mathcal{H}^{1st,Roe}$. To improve accuracy of the scheme no entropy fix (3.34) is applied.

Farfield boundary condition. The implementation of this boundary condition determines the inflow of free-stream conditions of the fluid. Given an angle of attack α, the outer state is determined by

$$W_{i,\text{bdry}} := W_\infty := (\rho_\infty, \cos\alpha\rho_\infty u_\infty, 0, \sin\alpha\rho_\infty u_\infty, \rho_\infty E_\infty)^T.$$

The determination of free-stream conditions for the turbulent flow equations is a complex topic and not straightforward. Here we simply give the values used, which are

$$\tilde{\nu}_{i,\text{bdry}} := \tilde{\nu}_\infty := f_{\text{SA},\infty}\tilde{\nu}_{l,\infty}, \qquad 3 \le f_{\text{SA},\infty} \le 5,$$

where we have followed the recommendations in [81, 1]. The constant factor $f_{\text{SA},\infty}$ determines the degree of turbulence in the free-stream and its choice depends for example on the simulated flow itself, the Reynolds number and the size of the finite mesh. Here one often counts the chord lengths from the simulated body to the free stream. As a result, the choice above yields for the Spalart-Allmaras turbulence model a ratio of eddy to laminar viscosity by

$$\frac{\mu_{t,\infty}}{\mu_{l,\infty}} = \frac{f_{\text{SA},\infty}\rho_\infty\tilde{\nu}_{l,\infty} f_{v1}(f_{\text{SA},\infty}\tilde{\nu}_{l,\infty}, W_\infty)}{\rho_\infty\tilde{\nu}_{l,\infty}} = \frac{f_{\text{SA},\infty}^4}{f_{\text{SA},\infty}^3 + c_{v1}^3}.$$

In all our computations for the Spalart-Allmaras turbulence model we chose

$$f_{\text{SA},\infty} = 3,$$

which goes along the recommendations given in [81, 1] and gives a ratio of about

$$\frac{\mu_{t,\infty}}{\mu_{l,\infty}} \approx 0.21044.$$

For the $k\omega$-model in all our computations the free-stream values are given by

$$(k_{i,\mathrm{bdry}}, \omega_{i,\mathrm{bdry}}) := (k_\infty, \omega_\infty) := \left(9 \cdot 10^{-9}, 10^{-6}\right).$$

Hence, this choice gives using (2.54) a ratio of eddy to laminar viscosity by

$$\frac{\mu_{t,\infty}}{\mu_{l,\infty}} = \frac{\frac{\sqrt{\gamma} M_\infty L}{Re} \rho \frac{k_\infty}{\omega_\infty}}{\frac{\sqrt{\gamma} M_\infty L}{Re}} = 9 \cdot 10^{-3}.$$

It is important to notice that this ratio of eddy to laminar viscosity for the $k\omega$-model is about two orders of magnitude smaller than compared with the Spalart-Allmaras turbulence model. Nevertheless, the mathematical question if these boundary conditions are meaningful as closure for the corresponding boundary value problem is open.

Slip wall boundary condition. The slip wall boundary conditions corresponds to the inviscid flow problem. To ensure a vanishing normal velocity the outer state is determined by

$$\begin{aligned} \rho_{i,\mathrm{bdry}} &:= \rho_i, \\ (\rho u)_{i,\mathrm{bdry}} &:= (\rho u)_i - 2 \left\langle (\rho u)_i, n_{i,\mathrm{bdry}} \right\rangle n_{i,\mathrm{bdry}}, \\ \rho_{i,\mathrm{bdry}} E_{i,\mathrm{bdry}} &:= \rho_i E_i. \end{aligned}$$

For results presented in this thesis this boundary condition was only used for inviscid flows. Hence, a possible treatment for turbulence flow equations is not considered.

No-slip wall boundary condition. The no-slip wall boundary corresponds to the viscous and turbulent flow problems. To ensure vanishing velocity and normal derivative of the temperature the outer state is determined by

$$\rho_{i,\mathrm{bdry}} := \rho_i, \tag{3.82a}$$
$$(\rho u)_{i,\mathrm{bdry}} := -(\rho u)_i, \tag{3.82b}$$
$$(\rho E)_{i,\mathrm{bdry}} := (\rho E)_i. \tag{3.82c}$$

The vanishing normal derivative

$$\left.\frac{\partial T}{\partial n}\right|_{e_{i,\mathrm{bdry}}} = 0 \tag{3.83}$$

is directly incorporated into the flux such that an application to (2.8b) yields

$$\left\langle q\left(W_i, W_{j,j\in\mathcal{N}(i)}\right), n_{i,\mathrm{bdry}} \right\rangle = \kappa_{\mathrm{eff},e_{i,\mathrm{bdry}}} \left.\frac{\partial T}{\partial n}\right|_{e_{i,\mathrm{bdry}}} = 0.$$

To determine the derivatives of velocity u, and the turbulent flow variables $\tilde{\nu}$, k and ω four methods are suggested. To this end consider the model of a polyhedral

mesh shown in Figure 3.3. Assuming that for laminar and turbulent flows the computational mesh is generated such that the boundary layer consists of quadrilateral elements in 2D or hexahedral elements in 3D such as on the right of Figure 3.3, the point $p_{i,n}$ can be used to approximate the derivative using (3.65). Assuming a linear behavior of u, $\tilde{\nu}$ and k over the boundary edge $e_{i,\text{bdry}}$, we define

$$u_{i,\text{bdry}} := -u_{i,n}, \tag{3.84a}$$
$$\tilde{\nu}_{i,\text{bdry}} := -\tilde{\nu}_{i,n}, \tag{3.84b}$$
$$k_{i,\text{bdry}} := -k_{i,n}, \tag{3.84c}$$

to approximate derivatives for the boundary edge by

$$\left(\frac{\partial u}{\partial x_k}\right)^{\text{TSL}}_{e_{i,\text{bdry}}} := \left.\frac{(n_{i,\text{bdry}})_k\,(u_{i,\text{bdry}} - u_i)}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}\right|_{u_i=0} = \frac{-(n_{i,\text{bdry}})_k\,u_{i,n}}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}, \tag{3.85a}$$
$$\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)^{\text{TSL}}_{e_{i,\text{bdry}}} := \left.\frac{(n_{i,\text{bdry}})_k\,(\tilde{\nu}_{i,\text{bdry}} - \tilde{\nu}_i)}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}\right|_{\tilde{\nu}_i=0} = \frac{-(n_{i,\text{bdry}})_k\,\tilde{\nu}_{i,n}}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}, \tag{3.85b}$$
$$\left(\frac{\partial k}{\partial x_k}\right)^{\text{TSL}}_{e_{i,\text{bdry}}} := \left.\frac{(n_{i,\text{bdry}})_k\,(k_{i,\text{bdry}} - k_i)}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}\right|_{k_i=0} = \frac{-(n_{i,\text{bdry}})_k\,k_{i,n}}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}. \tag{3.85c}$$

The determination of the boundary derivatives (3.85a)–(3.85c) have the major drawback that knowledge of $p_{i,n}$ is required. For example, assuming non-quadrilateral elements in the neighborhood of a no-slip wall these approximations are maybe misleading. Such a situation is illustrated in Figure 3.3 (left). This suggests that alternative methods are required. Hence, we can replace (3.84a)–(3.84c) by

$$u_{i,\text{bdry}} := -u_i, \tag{3.86a}$$
$$\tilde{\nu}_{i,\text{bdry}} := -\tilde{\nu}_i, \tag{3.86b}$$
$$k_{i,\text{bdry}} := -k_i. \tag{3.86c}$$

The settings (3.86a)–(3.86c) are consistent with (3.82b). Then, the approximate derivatives for the boundary edge may be computed by

$$\left(\frac{\partial u}{\partial x_k}\right)^{\text{TSL}}_{e_{i,\text{bdry}}} := \frac{(n_{i,\text{bdry}})_k\,(u_{i,\text{bdry}} - u_i)}{\|p_{i,\text{bdry}} - p_{i,n}\|_2} = \frac{-2\,(n_{i,\text{bdry}})_k\,u_i}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}, \tag{3.87}$$
$$\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)^{\text{TSL}}_{e_{i,\text{bdry}}} := \frac{(n_{i,\text{bdry}})_k\,(\tilde{\nu}_{i,\text{bdry}} - \tilde{\nu}_i)}{\|p_{i,\text{bdry}} - p_{i,n}\|_2} = \frac{-2\,(n_{i,\text{bdry}})_k\,\tilde{\nu}_i}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}, \tag{3.88}$$
$$\left(\frac{\partial k}{\partial x_k}\right)^{\text{TSL}}_{e_{i,\text{bdry}}} := \frac{(n_{i,\text{bdry}})_k\,(k_{i,\text{bdry}} - k_i)}{\|p_{i,\text{bdry}} - p_{i,n}\|_2} = \frac{-2\,(n_{i,\text{bdry}})_k\,k_i}{\|p_{i,\text{bdry}} - p_{i,n}\|_2}. \tag{3.89}$$

The computations (3.87)–(3.89) still require the distance $\|p_{i,\text{bdry}} - p_{i,n}\|_2$. To find a formulation which is free of this additional value we may apply (3.60) to determine

the derivative in the boundary cell

$$\left(\frac{\partial u}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} = \frac{1}{\mathrm{vol}\,(D_i)}\left[\sum_{j\in\mathcal{N}(i)} \mathrm{svol}(e_{ij})\frac{n_{k,ij}}{2}\,(u_i+u_j)|_{u_i=u_{i,\mathrm{bdry}}} + \mathrm{svol}(e_{i,\mathrm{bdry}})\frac{(n_{i,\mathrm{bdry}})_k}{2}\,(u_i+u_{i,\mathrm{bdry}})|_{u_i=u_{i,\mathrm{bdry}}}\right], \tag{3.90}$$

$$\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} = \frac{1}{\mathrm{vol}\,(D_i)}\left[\sum_{j\in\mathcal{N}(i)} \mathrm{svol}(e_{ij})\frac{n_{k,ij}}{2}\,(\tilde{\nu}_i+\tilde{\nu}_j)|_{\tilde{\nu}_i=\tilde{\nu}_{i,\mathrm{bdry}}} + \mathrm{svol}(e_{i,\mathrm{bdry}})\frac{(n_{i,\mathrm{bdry}})_k}{2}\,(\tilde{\nu}_i+\tilde{\nu}_{i,\mathrm{bdry}})|_{\tilde{\nu}_i=\tilde{\nu}_{i,\mathrm{bdry}}}\right], \tag{3.91}$$

$$\left(\frac{\partial k}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} = \frac{1}{\mathrm{vol}\,(D_i)}\left[\sum_{j\in\mathcal{N}(i)} \mathrm{svol}(e_{ij})\frac{n_{k,ij}}{2}\,(k_i+k_j)|_{k_i=k_{i,\mathrm{bdry}}} + \mathrm{svol}(e_{i,\mathrm{bdry}})\frac{(n_{i,\mathrm{bdry}})_k}{2}\,(k_i+k_{i,\mathrm{bdry}})|_{k_i=k_{i,\mathrm{bdry}}}\right]. \tag{3.92}$$

As above, in principal we can choose for the boundary values $u_{i,\mathrm{bdry}}, \tilde{\nu}_{i,\mathrm{bdry}}$ and $k_{i,\mathrm{bdry}}$ either (3.84a)–(3.84c) or (3.86a)–(3.86c) or

$$u_{i,\mathrm{bdry}} := 0, \tag{3.93a}$$
$$\tilde{\nu}_{i,\mathrm{bdry}} := 0, \tag{3.93b}$$
$$k_{i,\mathrm{bdry}} := 0. \tag{3.93c}$$

So, altogether as mentioned above, four methods are available to approximate the required derivatives. It is open which of the methods might be preferred, and this thesis will not investigate this topic in detail. Within the given implementation all methods were implemented and tested. For the examples considered in this thesis, results were similar, but a thorough study has not been performed. For all the results shown in Chapter 6 either (3.85a)–(3.85c) or (3.90)–(3.92) in combination with (3.93a)–(3.93c) has been used.

To realize the boundary condition for ω given by (2.55), note that when approaching a smooth no-slip wall, the asymptotic behavior is determined by (see [113])

$$\lim_{h\to 0+} \omega\,(x-hn(x)) = \lim_{h\to 0+} \frac{6\nu_l\,(W\,(x-hn(x)))}{\beta_{\mathrm{no-slip}}\,\|x-hn(x)\|_2^2}, \qquad \beta_{\mathrm{no-slip}} := \frac{9}{100}.$$

For the implementation first the term in the denominator is approximated by

$$\|x-hn(x)\|_2 \approx d_i \approx \|p_{i,\mathrm{bdry}} - p_{i,n}\|\,,$$

where d_i is the distance to the closest wall of the next discrete point. Then, to resolve within a discrete implementation the asymptotic behavior it was suggested [59, 61] to introduce an additional factor of 10, that is ω on the no-slip wall needs to satisfy

$$\omega_{\mathrm{no-slip}}(x) = \frac{60\nu_l(W(x))}{\beta_{\mathrm{no-slip}}\|p_{i,\mathrm{bdry}} - p_{i,n}\|_2^2}, \qquad x \in \partial D. \tag{3.94}$$

Based on the formulation (3.94), to find a suitable value $\omega_{i,\mathrm{bdry}}$ we suggest the following extrapolation. The gradient in normal direction is evaluated by

$$\left(\frac{\partial\omega}{\partial x_k}\right)_{e_{i,\mathrm{bdry}}}^{\mathrm{TSL},app} \approx \frac{(n_{i,\mathrm{bdry}})_k(\omega_{\mathrm{no-slip}} - \omega_{i,n})}{\|p_{i,\mathrm{bdry}} - p_{i,n}\|_2}.$$

This approximation is exploited to determine

$$\begin{aligned}\omega_{i,\mathrm{bdry}} &:= \omega_{\mathrm{no-slip}} + f_\omega\left\langle\left(\frac{\partial\omega}{\partial x}\right)_{e_{i,\mathrm{bdry}}}^{\mathrm{TSL},app}, n_{i,\mathrm{bdry}}\right\rangle \\ &= \omega_{\mathrm{no-slip}} + f_\omega\frac{(\omega_{\mathrm{no-slip}} - \omega_{i,n})}{\|p_{i,\mathrm{bdry}} - p_{i,n}\|_2}, \qquad f_\omega := 4.\end{aligned}$$

This is exactly the value we used for ω on the no-slip wall in all our computations for the $k\omega$-model. Then, the approximate derivative for the boundary edge required for the viscous terms is given by

$$\left(\frac{\partial\omega}{\partial x_k}\right)_{e_{i,\mathrm{bdry}}}^{\mathrm{TSL}} := \frac{(n_{i,\mathrm{bdry}})_k(\omega_{i,\mathrm{bdry}} - \omega_i)}{\|p_{i,\mathrm{bdry}} - p_{i,n}\|_2}.$$

Note that in principle we could have considered for this construction also Green-Gauss gradients. For the sake of simplicity we restricted ourselves to the presented construction. It turned out that in particular the reconstruction of a suited boundary value $\omega_{\mathrm{no-slip}}$ is within our implementation responsible for a robust discretization and realization of a solution method, which has the potential to solve the set of equations of the $k\omega$-model.

There is a further important remark. When solving for ω we scaled this variable by

$$\omega_{sc} = \frac{Re}{\sqrt{\gamma}M_\infty L}.$$

Due to the boundary condition (2.55) which is approximated by (3.94) it is clear that with respect to possible mesh refinements the distance to the closest wall

$$d_{h,i} \approx \|p_{i,\mathrm{bdry}} - p_{h,i,n}\|_2 \to 0, \qquad h \to 0,$$

and hence possibly numerical instabilities emerge due to a significant increase in the boundary values $\omega_{\mathrm{no-slip}}$. To resolve the flow inside the boundary layer meshes are

in general generated such that this distance scales with the Reynolds number, $d_i = d_i(Re) \sim C/Re$. Hence, using (2.10), (2.13) and (2.31a) we conclude that at least up to some mesh refinement that scaling ω_{sc} leads to a normalization of the possibly largest values expected, that is

$$\omega_{\text{no-slip}} \sim \tilde{C} \frac{1}{\omega_{sc}} \frac{\frac{\sqrt{\gamma} M_\infty L}{Re}}{\left(\frac{C}{Re}\right)^2} \sim \overline{C}.$$

In case ω_{sc} would be chosen by one, we may conclude that the $\omega_{\text{no-slip}}$ already behaves like the Reynolds number Re in an neighborhood of the no-slip wall. And these values might be large for high Reynolds number turbulent flows. Hence, the scaling ω_{sc} only turns out to be a typical normalization for the variable ω, which may stabilize the numerical implementation.

Symmetry boundary condition. A further boundary condition considered and implemented is the symmetry boundary condition. This boundary condition is similar to the slip wall boundary. Here all vector valued quantities need to be projected, that is not only vector valued variables but also possible gradients. For completeness, we recall the requirement

$$\begin{aligned} \rho_{i,\text{bdry}} &:= \rho_i, \\ (\rho u)_{i,\text{bdry}} &:= (\rho u)_i - 2 \left\langle (\rho u)_i, n_{i,\text{bdry}} \right\rangle n_{i,\text{bdry}}, \\ \rho_{i,\text{bdry}} E_{i,\text{bdry}} &:= \rho_i E_i, \end{aligned}$$

together with possible quantities of the turbulence flow equations,

$$\begin{aligned} \tilde{\nu}_{i,\text{bdry}} &= \tilde{\nu}_i, \\ k_{i,\text{bdry}} &= k_i, \\ \omega_{i,\text{bdry}} &= \omega_i. \end{aligned}$$

Assuming that $v_i \in \mathbb{R}^3$ is any other vector valued quantity, such as a gradient, we define its boundary value by

$$v_{i,\text{bdry}} := v_i - 2 \left\langle v_i, n_{i,\text{bdry}} \right\rangle n_{i,\text{bdry}}.$$

Then these values are incorporated into the flux functions for the symmetry boundary. This consideration concludes the implementation details of boundary conditions. Note that these techniques are not restricted to finite volume codes, but the same ideas can be applied to for example to discontinuous Galerkin codes.

3.7 Discrete set of equations

Using the notation of (3.9) and (3.10) to formulate the complete set of equations, we define the coefficient vector

$$\mathbf{W}_{\text{mean}}(t) := (W_1(t), \ldots, W_{N_{\text{elem}}}(t))$$

to represent the ansatz for function for the mean flow equation (2.45). For the turbulence flow equation we use in the following the same notation for both the Spalart-Allmaras and the Wilcox $k\omega$-model. Considering the representation of the ansatz functions $\tilde{\nu}_h$ and (k_h, ω_h) introduced in Section 3.5 we denote the corresponding coefficient vector either by

$$\mathbf{W}_t(t) := (\tilde{\nu}_1(t), \ldots, \tilde{\nu}_{N_{\text{elem}}}(t))$$

or

$$\mathbf{W}_t(t) := ((k_1(t), \omega_1(t)), \ldots, (k_{N_{\text{elem}}}(t), \omega_{N_{\text{elem}}}(t))).$$

Then, the discretization of the mean flow equations (2.45) together with the turbulent flow equations (2.52) or (2.53) yield the system of ordinary differential equations

$$\frac{d}{dt}\begin{pmatrix} \mathbf{W}(t) \\ \mathbf{W}_t(t) \end{pmatrix} = \begin{pmatrix} -\mathbf{M}_{\text{mean}}^{-1}\mathbf{R}_{\text{mean}}(\mathbf{W}(t), \mathbf{W}_t(t)) \\ -\mathbf{M}_{\text{turb}}^{-1}\mathbf{R}_{\text{turb}}(\mathbf{W}(t), \mathbf{W}_t(t)) \end{pmatrix}, \tag{3.95}$$

where $\mathbf{M}_{\text{mean}} := \text{diag}(\text{diag}(\text{vol}(D_i))) \in \mathbb{R}^{5N_{\text{elem}} \times 5N_{\text{elem}}}$ and $\mathbf{M}_{\text{turb}} := \text{diag}(\text{vol}(\Omega_i)) \in \mathbb{R}^{N_t \cdot N_{\text{elem}} \times N_t \cdot N_{\text{elem}}}$ denote the mass matrix for mean and turbulent flow equations. Depending on the actual mesh size, (3.95) represents a large scale, time dependent set of nonlinear equations which need to be iterated in time. To approximately solve (3.95) we assume that the mean flow equations depend only on $\mathbf{W}$ and $\mathbf{W}_t$ acts only as parameter here, whereas the turbulence flow equation depend only on $\mathbf{W}_t$ and $\mathbf{W}$ act as parameter. Hence, we rewrite system (3.95) as

$$\frac{d}{dt}\mathbf{W}(t) = -\mathbf{M}_{\text{mean}}^{-1}\mathbf{R}_{\text{mean}}(\mathbf{W}(t); \mathbf{W}_t(t)) \tag{3.96a}$$

$$\frac{d}{dt}\mathbf{W}_t(t) = -\mathbf{M}_{\text{turb}}^{-1}\mathbf{R}_{\text{turb}}(\mathbf{W}_t(t); \mathbf{W}(t)). \tag{3.96b}$$

Equations (3.96a) and (3.96b) are then solved sequentially. Mentioned in Section 2.4 it is not our goal to approximate time accurate solutions of (3.95), but our main interest is the robust approximation of a steady state solution. That is we postulate that the left hand side of (3.95) vanishes, that is it satisfies

$$\frac{d}{dt}\mathbf{W}(t) = 0, \qquad \frac{d}{dt}\mathbf{W}_t(t) = 0.$$

With respect to this assumption the system (3.96) simplifies to

$$0 = \mathbf{R}_{\text{mean}}(\mathbf{W}(t); \mathbf{W}_t(t)) \tag{3.97a}$$

$$0 = \mathbf{R}_{\text{turb}}(\mathbf{W}_t(t); \mathbf{W}(t)), \tag{3.97b}$$

which represents a nonlinear set of equations which needs to be solved. In principle, Newton's method is suggested to be the straightforward way to solve this set of equations. Based on this observation, it is the goal of Section 5 to suggest solution strategies.

Chapter 4

Total derivative of Discretization

The computation of the derivative of the residual $\mathbf{R}$ is complex. Therefore, a full chapter is devoted for this section. In principle, the process to construct the derivative $d\mathbf{R}/d\mathbf{W}$ can be implemented in a fully automatic fashion and only the local fluxes over the faces e_{ij} and the source terms which are given pointwise need to be differentiated. Then the complete derivative is constructed in the same way the residual $\mathbf{R}$ itself is computed. This basic and simple observation suggests to distribute this chapter into two parts.

The first part describes the consequence of this global observation and it presents the required data structures and routines to construct the derivative matrix. In particular, note that such an approach is independent of the underlying integral equation or differential equations. It depends only on the actual stencils used for the discretization. Due to this fact amongst other things we have spent so much effort in Chapter 3 to identify the dependencies on the ansatz functions and corresponding coefficients.

In the second part of this chapter, we present the derivatives of the fluxes and source terms with respect to their discretization. On the one hand, we present for several terms the complete derivatives to show in which way an exact derivative of the residual can be constructed. From the terms shown it may become clear that this is a rather extensive analysis with lots of details. Without getting lost in details, we are not going to present this for all terms. The goal is to make clear what needs to be done to compute these terms and in which way to incorporate those. Hence, in particular for the viscous flux terms we restrict ourselves to those terms which are necessary for the solution method presented in Chapter 5. Nevertheless, for future work and readers generalizations to construct exact derivatives beyond those terms presented in this chapter should become clear. Such an exact derivative is for example important for optimization problems, control problems and inverse problems based on the boundary value problems considered in Section 2.4.

For understanding the data structures and the implemented algorithms, within this work a private implementation dealing with block and scalar sparse matrices was

accomplished. No other software has been used. Though, obviously this meant that from a software design point of view there is still tremendous potential for efficiency, it is the goal of this work to demonstrate the potential of suitable algorithms to solve the RANS equations. With this in mind, it was an intentional decision to include software written by others to a minimum.

Moreover, our major goal is the design of an efficient solution algorithm to approximate a solution to the boundary value problem formulated in Section 2.4. Such a method is realized introducing preconditioners which are based on certain simplifications of the exact derivative. These simplifying assumptions are discussed in this chapter as well.

4.1 Global considerations

This section discusses the structure of $\frac{d\mathbf{R}}{d\mathbf{W}}$. In general, this structure is at the outset independent of the actual integral or differential equations, but it depends on the stencil used for discretization. Hence, this expression represents a large scale (block) sparse matrix, which needs to be saved efficiently in the fast memory of a computer such that the available resources are not overloaded. Since we want to make extensive use in our solution method, we start by considering a suitable representation.

4.1.1 Graphs and sparse matrices

For this section recall the notation of Section 3.1. To understand the structure of the full derivative $\frac{d\mathbf{R}}{d\mathbf{W}}$ it was one goal of the last Section 3 to emphasize the dependency of $\mathbf{R}_i$ on the unknowns $\mathbf{W}$. To represent the (block) sparse matrices a **C**ompressed **S**parse **R**ow format has been implemented. For details of this topic we refer to the text book of Saad [84]. For our purposes we shortly introduce the following definition and notation.

Definition 4.1.1 *The neighbors of neighbors of point i are defined by*

$$\mathcal{N}\left(\mathcal{N}\left(i\right)\right) := \left(\cup_{k\in\mathcal{N}(i)}\mathcal{N}\left(k\right)\right) \setminus \left(\left(\cap_{k\in\mathcal{N}(i)}\mathcal{N}\left(k\right)\right)\right).$$

Notation 4.1.2 *We denote the neighbors and neighbors of neighbors of point i in the following explicitly by*

$$\begin{aligned} \mathcal{N}(i) &= \left\{j_1,\ldots,j_{\#\mathcal{N}(i)}\right\}, \\ \mathcal{N}\left(\mathcal{N}\left(i\right)\right) &= \left\{k_1,\ldots,k_{\#\mathcal{N}(\mathcal{N}(i))}\right\}. \end{aligned}$$

Corollary 4.1.3 *Consider the functions* $\mathbf{R}_{\text{mean}}$ *and* $\mathbf{R}_{\text{turb}}$ *of Section 3.7. The* i*th component* $(\mathbf{R}_{\text{mean}})_i$ *and* $(\mathbf{R}_{\text{turb}})_i$ *are a function of the variables of the indices* i, $\mathcal{N}(i)$ *and* $\mathcal{N}(\mathcal{N}(i))$, *that is*

$$(\mathbf{R}_{\text{mean}})_i(\mathbf{W}) = (\mathbf{R}_{\text{mean}})_i\left(W_i, W_{j_1}, \ldots, W_{j_{\#\mathcal{N}(i)}}, W_{k_1}, \ldots, W_{k_{\#\mathcal{N}(\mathcal{N}(i))}}\right), \quad (4.1a)$$

$$(\mathbf{R}_{\text{turb}})_i(\mathbf{W}_t) = (\mathbf{R}_{\text{mean}})_i\left(\left(W_i, W_{j_1}, \ldots, W_{j_{\#\mathcal{N}(i)}}, W_{k_1}, \ldots, W_{k_{\#\mathcal{N}(\mathcal{N}(i))}}\right)_t\right). \quad (4.1b)$$

Proof: This is a consequence of the construction of the residual, which is described in Section 3.

□

It is exactly these indices i, $\mathcal{N}(i)$ and $\mathcal{N}(\mathcal{N}(i))$ which determine the number of nonzero entries and the column indices in the ith row of the block sparse matrix. More general, we can interpret these connections as a graph G, and the structure of the (block) sparse matrix representing the derivative $\frac{d\mathbf{R}}{d\mathbf{W}}$ is determined by the points which are connected by an edge in the graph. Considering the set $\mathcal{N}(\mathcal{N}(i))$ there exist also edges in this graph which do not correspond to the directly connected edges given by interpretation as a mesh. To get an idea about the memory requirements to save $\frac{d\mathbf{R}}{d\mathbf{W}}$ in the fast memory of a computer we give the following lemma.

Lemma 4.1.4 *Consider a 3D hexahedral structured mesh representing* N *degrees of freedom. Then the memory requirements for* $\frac{d\mathbf{R}_{\text{mean}}}{d\mathbf{W}}$ *are*

$$5000 \cdot N Bytes.$$

Proof: For a 3D hexahedral mesh we have

$$\#\{i, j \in \mathcal{N}(i), k \in \mathcal{N}(\mathcal{N}(i))\} = 25. \quad (4.2)$$

Hence, each row of $\frac{d\mathbf{R}}{d\mathbf{W}}$ has 25 nonzero entries. For the discretized mean flow equations each entry is a block with $5 \cdot 5 = 25$ entries. Considering the programming languages C [34] or C++ [87], where a double requires 8 bytes, the memory requirements for the matrix are

$$25 \cdot 25 \cdot 8 \cdot N\text{bytes} = 5000N\text{Bytes}.$$

□

For example, for a mesh with $N = 10^6$ nodes about $5GB$ memory are required to save the matrix $\frac{\partial \mathbf{R}_{\text{mean}}}{\partial \mathbf{W}}$. Modern hardware clusters in general have much more memory available.

To construct efficient solution algorithms, which is topic of the next Chapter 5, the construction of the exact derivative $\frac{d\mathbf{R}}{d\mathbf{W}}$ often plays a minor role. Here it is of major importance to find suitable approximations to the exact derivative such that these can be incorporated as preconditioners into the solution algorithm. To this end we introduce the following notation.

Notation 4.1.5 *Assume that the ith component of the residual, that is* $\mathbf{R}_i$*, depends only on* W_i *and its direct neighbors* W_j*,* $j \in \mathcal{N}(i)$*. Then we call the dependency a compact stencil and denote this by* $\mathbf{R}_i^{\text{comp}}$*.*

For a compact residual $\mathbf{R}_i^{\text{comp}}$ the nonzero entries in the (block) CSR matrix are determined by i and $\mathcal{N}(i)$. To construct this matrix a single loop over all edges is enough, that is Algorithm 2 below reduces to the first loop over all edges. Furthermore, we can give a rough estimate for the memory requirements.

Lemma 4.1.6 *Consider a 3D hexahedral structured mesh representing* N *degrees of freedom. Then the memory requirements for* $\frac{d\mathbf{R}_{\text{mean}}^{\text{comp}}}{d\mathbf{W}}$ *are*

$$1400 \cdot N Bytes.$$

Proof: The proof follows line by line the proof of Lemma 4.2 taking into account that (4.2) needs to be exchanged by

$$\#\{i, j \in \mathcal{N}(i)\} = 7.$$

□

Hence, the memory requirement to store (block) CSR matrices for a compact residual $\frac{d\mathbf{R}_{\text{mean}}^{\text{comp}}}{d\mathbf{W}}$ is about 3.57 times less than storing the matrices $\frac{d\mathbf{R}_{\text{mean}}}{d\mathbf{W}}$ for a residual requiring an extended stencil.

4.1.2 Construction of $\frac{d\mathbf{R}}{d\mathbf{W}}$

For the construction of $\frac{d\mathbf{R}}{d\mathbf{W}}$ it is important to distinct on the one hand between flux terms depending only on variables W_i and W_j, that is those directly corresponding to the edge (face) e_{ij}, and on the other hand those being additionally dependent on the neighbors of these variables, that is those which depend on W_i and W_j and additionally on W_k, $k \in \mathcal{N}(i)$ and W_ℓ, $\ell \in \mathcal{N}(j)$. We emphasize this point because our way to implement the derivative follows three major goals:

a) The technical realization to compute $\frac{d\mathbf{R}}{d\mathbf{W}}$ should be as close as possible to the evaluation of the residual $\mathbf{R}$.

b) The stencil operations, that is the generation of the structure of the matrix, should be done (almost) automatically.

c) Only the fluxes over the edges need to be differentiated, as well as volume operations for the source terms.

This approach is important. In general a software package implementing the RANS or any other differential or integral equations is not fixed, but under continuous change, and it is not only one person manipulating the code, but many. In case one has separated the technical stencil operations from the actual discretization, developers of the code need to care only about differentiation of their local stencil operations. The global interaction is realized automatically and is therefore free from defects. In a final step, one could even realize the local differentiation by automatic differentiation. At the final end, this might be a question of efficiency versus degree of automation.

To realize the global interaction, we need to resolve the dependencies of $\mathbf{R}_i$ on the variables $\mathbf{W}$. Mathematically spoken, we need to resolve the dependencies, which is realized exploiting the chain rule for differentiation. Assuming that we represent the reconstructions required for the evaluation of $\mathbf{R}$ by a function $g = g(\mathbf{W})$, we need to understand what the expression

$$\frac{d\mathbf{R}}{d\mathbf{W}} = \frac{d\mathbf{R}}{dg}\frac{dg}{d\mathbf{W}}$$

means, in particular such that a realizable way to implement it into a computer code can be found. It is our goal to describe an automatic process to realize this technical challenge. Then, within such a generalized framework, the detailed derivatives of certain terms can be easily exchanged.

To describe the idea in the following, consider the ith component of the residual and the corresponding edges (faces) e_{ij}, $j \in \mathcal{N}(i)$. Then the corresponding residual is computed by

$$\mathbf{R}_i = \sum_{j \in \mathcal{N}(i)} \mathrm{svol}(e_{ij}) \left[\mathcal{H}\left(W_{\mathcal{N}(i,j)}, n_{e_{ij}}\right) - \left\langle f_v\left(W_{\mathcal{N}(i,j)}\right), n_{e_{ij}} \right\rangle \right].$$

Note that within a face based code we do not directly evaluate $\mathbf{R}_i$, $i = 1, \ldots, N_{elem}$, but to save operational count the terms over the edges are computed and summed up. This approach is summarized in Algorithm 1. Here it is documented, in which way the extended stencil operations are realized, namely in a sequential manner. It is exactly this sequential progression which needs to be mapped to the construction of $\frac{d\mathbf{R}}{d\mathbf{W}}$.

In principle the explicit dependencies mentioned in Corollary 4.1.3 in (4.1) can be directly resolved to compute the derivatives. But for efficiency, and due to the reasons discussed in the beginning of this section, in general the realization of the computation of the residual is done by looping over the edges. To carry this concept over to the computation of the derivative, we consider the residual as dependent only on reconstructed values (Laplacians and gradients) corresponding to the direct neighbors, that is (4.1) is rewritten by

$$\mathbf{R}_i = \mathbf{R}_i\left(g(W_i), g_{j_1}\left(W_{k,k\in\mathcal{N}(j_1)}\right), \ldots, g_{j_{\#\mathcal{N}(i)}}\left(W_{k,k\in\mathcal{N}\left(j_{\#\mathcal{N}(i)}\right)}\right)\right). \tag{4.3}$$

Algorithm 1 Computation of residual

procedure LOOP OVER EDGES (FACES) TO COMPUTE GRADIENTS
  for $k = 1, \ldots, \#E(M)$ **do**
    $i \leftarrow e_{ij}$
    $j \leftarrow e_{ij}$
    $L_i \leftarrow L_i + (W_j - W_i)$
    $L_j \leftarrow L_j - (W_j - W_i)$
    $\left(\frac{\partial u}{\partial x}\right)_{D_i} \leftarrow \left(\frac{\partial u}{\partial x}\right)_{D_i} + \frac{\mathrm{svol}(e_{ij})}{2}(u_i + u_j)\, n_{e,ij}$
    $\left(\frac{\partial T}{\partial x}\right)_{D_i} \leftarrow \left(\frac{\partial T}{\partial x}\right)_{D_i} + \frac{\mathrm{svol}(e_{ij})}{2}(T_i + T_j)\, n_{e,ij}$
procedure LOOP OVER ELEMENTS FINISHING GRADIENTS
  for $k = 1, \ldots, N_{\mathrm{elem}}$ **do**
    $\left(\frac{\partial u}{\partial x}\right)_{D_i} \leftarrow (\mathrm{vol}(D_k))^{-1} \left(\frac{\partial u}{\partial x}\right)_{D_k}$
    $\left(\frac{\partial T}{\partial x}\right)_{D_i} \leftarrow (\mathrm{vol}(D_k))^{-1} \left(\frac{\partial T}{\partial x}\right)_{D_k}$
procedure LOOP OVER EDGES (FACE) TO COMPUTE RESIDUUM
  for $k = 1, \ldots, \#E(M)$ **do**
    $i \leftarrow e_{ij}$
    $j \leftarrow e_{ij}$
    $\mathcal{H}_{ij} \leftarrow \mathcal{H}(W_i, W_j, L_i, L_j, n)$
    $(f_v)_{ij} \leftarrow \left\langle f_v\left(W_i, W_j, \left(\frac{\partial u}{\partial x}\right)_{D_i}, \left(\frac{\partial u}{\partial x}\right)_{D_j}, \left(\frac{\partial T}{\partial x}\right)_{D_i}, \left(\frac{\partial T}{\partial x}\right)_{D_j}\right), n_{e_{ij}}\right\rangle$
    $\mathbf{R}_i \leftarrow \mathbf{R}_i + \mathrm{svol}(e_{ij})\left(\mathcal{H}_{ij} - (f_v)_{ij}\right)$
    $\mathbf{R}_j \leftarrow \mathbf{R}_j - \mathrm{svol}(e_{ij})\left(\mathcal{H}_{ij} - (f_v)_{ij}\right)$

		p_9		
	p_8	p_1	p_2	
p_{12}	p_7	p_0	p_3	p_{10}
	p_6	p_5	p_4	
		p_{11}		

Figure 4.1: Complete stencil for a 2D hexahedral mesh

Then, exploiting the chain rule the derivative is computed by

$$\frac{\partial \mathbf{R}_i}{\partial W_k} = \left(\frac{\partial \mathbf{R}_i}{\partial g}, \frac{\partial \mathbf{R}_i}{\partial g_1}, \ldots, \frac{\partial \mathbf{R}_i}{\partial g_{j_{\#\mathcal{N}(i)}}} \right) \begin{pmatrix} \frac{\partial g}{\partial W_k} \\ \frac{\partial g_1}{\partial W_k} \\ \vdots \\ \frac{\partial g_{j_{\#\mathcal{N}(i)}}}{\partial W_k} \end{pmatrix}. \tag{4.4}$$

Both, the representation of the $\mathbf{R}_i$ as function of $g, g_1, \ldots, g_{j_{\#\mathcal{N}(i)}}$ and the functions $g, g_1, \ldots, g_{j_{\#\mathcal{N}(i)}}$ depend only on their direct neighbors, and therefore, corresponding to the residual, the implementation of the derivative of the residual can be realized by a nested loop over the edges. And hence, only the direct local derivatives corresponding to one edge e_{ij} are required. Derivatives arising from an extended stencil follow by application of the chain rule.

We may present an easy example which can be computed straightforward. Considering a 2D hexahedral mesh, we need to represent the complete stencil. The example is illustrated in Figure 4.1. Using the notation of this example, the sum of differences in the definition of the numerical flux function $\mathcal{H}$ in (3.15) and (3.16) is

$$D_i\left(W_{\mathcal{N}(i,j)}\right) := \sum_{j \in \mathcal{N}(i)} \left(L_j\left(W_{\mathcal{N}(i,j)}\right) - L_i\left(W_{\mathcal{N}(i,j)}\right) \right). \tag{4.5}$$

A straightforward evaluation of this expression is

$$\begin{aligned} D_0\left(W_{\mathcal{N}(i,j)}\right) &= 20W_0 - 8\left(W_1 + W_3 + W_5 + W_7\right) \\ &\quad +2\left(W_2 + W_4 + W_6 + W_8\right) + W_9 + W_{10} + W_{11} + W_{12}. \end{aligned} \tag{4.6}$$

The derivatives $\frac{\partial D_0}{\partial W_i}$, $i = 0, \ldots, 12$ are obvious from (4.6). Such an explicit representation corresponds to (4.1). This simple example illustrates that such an explicit approach is in general and in particular for more complex formulae not feasible, and in particular not efficiently feasible.

To follow the idea of (4.3) the term D_0 is reformulated as

$$\begin{aligned} D_0\left(W_{\mathcal{N}(i,j)}\right) &= D_0\left(L_0, L_1, L_3, L_5, L_7\right), \\ L_i &= \sum_{j\in\mathcal{N}(i)} \left(W_j - W_i\right). \end{aligned}$$

Straightforward differentiation gives (we neglect the fact that the expressions need to be understood as matrix valued, the terms represent the diagonal of the matrix, off-diagonal terms are zero)

$$\frac{\partial D_0}{\partial\left(L_0, L_1, L_3, L_5, L_7\right)} = (-4, 1, 1, 1, 1).$$

Now, exploiting the chain rule (4.4) we can compute for example for the derivative with respect to W_0,

$$\frac{\partial D_0}{\partial W_0} = (-4,1,1,1,1)\begin{pmatrix} \frac{\partial L_0}{\partial W_0} \\ \frac{\partial L_1}{\partial W_0} \\ \frac{\partial L_3}{\partial W_0} \\ \frac{\partial L_5}{\partial W_0} \\ \frac{\partial L_7}{\partial W_0} \end{pmatrix} = (-4,1,1,1,1)\begin{pmatrix} -4 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = 20,$$

with respect to W_2,

$$\frac{\partial D_0}{\partial W_2} = (-4,1,1,1,1)\begin{pmatrix} \frac{\partial L_0}{\partial W_2} \\ \frac{\partial L_1}{\partial W_2} \\ \frac{\partial L_3}{\partial W_2} \\ \frac{\partial L_5}{\partial W_2} \\ \frac{\partial L_7}{\partial W_2} \end{pmatrix} = (-4,1,1,1,1)\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = 2,$$

with respect to W_3,

$$\frac{\partial D_0}{\partial W_3} = (-4,1,1,1,1)\begin{pmatrix} \frac{\partial L_0}{\partial W_3} \\ \frac{\partial L_1}{\partial W_3} \\ \frac{\partial L_3}{\partial W_3} \\ \frac{\partial L_5}{\partial W_3} \\ \frac{\partial L_7}{\partial W_3} \end{pmatrix} = (-4,1,1,1,1)\begin{pmatrix} 1 \\ 0 \\ -4 \\ 0 \\ 0 \end{pmatrix} = -8$$

and with respect to W_{10},

$$\frac{\partial D_0}{\partial W_{10}} = (-4,1,1,1,1)\begin{pmatrix} \frac{\partial L_0}{\partial W_{10}} \\ \frac{\partial L_1}{\partial W_{10}} \\ \frac{\partial L_3}{\partial W_{10}} \\ \frac{\partial L_5}{\partial W_{10}} \\ \frac{\partial L_7}{\partial W_{10}} \end{pmatrix} = (-4,1,1,1,1)\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = 1$$

And, naturally, these results correspond exactly to the result obtained by the explicit expression (4.6). To reformulate this idea into a loop over the edges (faces), we write

$$\begin{aligned}\frac{\partial D_i}{\partial W_k} &= \frac{\partial \sum_{j\in\mathcal{N}(0)}(L_j - L_i)}{\partial W_k} = \sum_{j\in\mathcal{N}(0)} \frac{\partial (L_j - L_i)}{\partial W_k} \\ &= \sum_{j\in\mathcal{N}(0)} \frac{\partial (L_j - L_i)}{\partial (L_i, L_j)} \left(\frac{\partial L_i}{\partial W_k}, \frac{\partial L_j}{\partial W_k}\right)^T \\ &= \sum_{e_{0j}, j\in\mathcal{N}(0)} \frac{\partial (L_j - L_i)}{\partial (L_i, L_j)} \left(\frac{\partial L_i}{\partial W_k}, \frac{\partial L_j}{\partial W_k}\right)^T .\end{aligned}$$

Algorithmically, the considerations above translate as follows:

Algorithm 2 Computation of derivative of Laplacians

procedure LOOP OVER EDGES (FACES) FOR INNER DERIVATIVES
  for $k = 1, \ldots, \#E(M)$ **do**
    $i \leftarrow e_{ij}$
    $j \leftarrow e_{ij}$
    $\frac{\partial L_i}{\partial W_i} \leftarrow \frac{\partial L_i}{\partial W_i} + \frac{\partial (W_j - W_i)}{\partial W_i}$
    $\frac{\partial L_i}{\partial W_j} \leftarrow \frac{\partial L_i}{\partial W_j} + \frac{\partial (W_j - W_i)}{\partial W_j}$
    $\frac{\partial L_j}{\partial W_i} \leftarrow \frac{\partial L_j}{\partial W_i} - \frac{\partial (W_j - W_i)}{\partial W_i}$
    $\frac{\partial L_j}{\partial W_j} \leftarrow \frac{\partial L_j}{\partial W_j} - \frac{\partial (W_j - W_i)}{\partial W_j}$
procedure LOOP OVER EDGES (FACES) TO COMPUTE FULL DERIVATIVE
  for $k = 1, \ldots, \#E(M)$ **do**
    $i \leftarrow e_{ij}$
    $j \leftarrow e_{ij}$
    $\frac{\partial D_i}{\partial W_i} \leftarrow \frac{\partial D_i}{\partial W_i} + \frac{\partial (L_j - L_i)}{\partial (L_i, L_j)} \left(\frac{\partial L_i}{\partial W_i}, \frac{\partial L_j}{\partial W_i}\right)^T ,$
    $\frac{\partial D_j}{\partial W_j} \leftarrow \frac{\partial D_j}{\partial W_j} - \frac{\partial (L_j - L_i)}{\partial (L_i, L_j)} \left(\frac{\partial L_i}{\partial W_j}, \frac{\partial L_j}{\partial W_j}\right)^T$
    for $\ell \in \mathcal{N}(i), \ell \neq i$ **do**
      $\frac{\partial D_i}{\partial W_j} \leftarrow \frac{\partial D_i}{\partial W_j} + \frac{\partial (L_\ell - L_i)}{\partial (L_i, L_\ell)} \left(\frac{\partial L_i}{\partial W_j}, \frac{\partial L_\ell}{\partial W_j}\right)^T$
      $\frac{\partial D_i}{\partial W_\ell} \leftarrow \frac{\partial D_i}{\partial W_\ell} + \frac{\partial (L_j - L_i)}{\partial (L_i, L_j)} \left(\frac{\partial L_i}{\partial W_\ell}, \frac{\partial L_j}{\partial W_\ell}\right)^T$
    for $\ell \in \mathcal{N}(j), \ell \neq j$ **do**
      $\frac{\partial D_j}{\partial W_i} \leftarrow \frac{\partial D_j}{\partial W_i} + \frac{\partial (L_\ell - L_j)}{\partial (L_j, L_\ell)} \left(\frac{\partial L_j}{\partial W_i}, \frac{\partial L_\ell}{\partial W_i}\right)^T$
      $\frac{\partial D_j}{\partial W_\ell} \leftarrow \frac{\partial D_j}{\partial W_\ell} - \frac{\partial (L_j - L_i)}{\partial (L_i, L_j)} \left(\frac{\partial L_i}{\partial W_\ell}, \frac{\partial L_j}{\partial W_\ell}\right)^T$

For simplicity we restricted within this section the example for the construction of the full derivative $\frac{d\mathbf{R}}{d\mathbf{W}}$ to the Laplacians D_i given in (4.5), which are part of $\mathcal{H}$ given in Definition 3.2.3. It was the intention that the implementation might be more obvious by explaining this construction using a straightforward example. All other terms which depend on the extended stencil corresponding to $\mathcal{N}(\mathcal{N}(i))$ can be realized following the same strategy. Only the inner and outer derivatives need to be exchanged. Then Algorithm 2 is a prototype to compute the complete derivatives. In the next section we are going to presents detailed descriptions of the derivatives for the flux terms over the edges (faces). The construction of the complete derivative is realized in an automatic fashion discussed in this section.

4.2 Derivative of inviscid terms

The computation of the derivative of the inviscid terms requires differentiation of the numerical flux function $\mathcal{H}$ given in Definition 3.2.3, that is

$$\frac{\partial \mathcal{H}}{\partial W_k} = \frac{\partial}{\partial W_k}\left\{\frac{1}{2}\left[\langle f_c(W_i), n\rangle + \langle f_c(W_j), n\rangle\right] - d_{ij}\left(W_{\mathcal{N}(i,j)}, n\right)\right\} \tag{4.7}$$

In this section we present the full derivative as well as simplified restrictions. Note that for evaluation of the derivatives we always assume $\mathbf{P}_{ij} = \mathbf{I}$. The derivative of the low Mach modified operators are not considered in this thesis. For the design of preconditioning techniques to construct efficient preconditioners we rely on the demand to efficiently evaluate only certain parts of the derivative of the inviscid part. This is also topic of this section.

4.2.1 Derivative of compact inviscid flux

To compute the derivative (4.7) in a first step we restrict ourselves to a compact approximation. Such a compact approximation is given by the flux function $\mathcal{H}^{1st,\mathrm{Roe}}$ of Definition 3.2.4.

For both $\mathcal{H}$ and $\mathcal{H}^{1st,\mathrm{Roe}}$ the derivatives of the convective fluxes are straightforward,

$$\frac{\partial \langle f_c(W_\ell), n\rangle}{\partial W_k} = \begin{cases} \frac{\partial \langle f_c, n\rangle[W_i]}{\partial W_i}, & k = \ell = i, \\ \frac{\partial \langle f_c, n\rangle[W_j]}{\partial W_j}, & k = \ell = j, \\ 0, & k \neq \ell. \end{cases} \tag{4.8}$$

These terms are evaluated using the explicit expression at the end of Section 3.2.1. The derivative $\frac{\partial d_{ij}}{\partial W_k}$ is far more complicated. Restricting ourselves to the flux function $\mathcal{H}^{1st,\mathrm{Roe}}$ of Definition 3.2.4, we compute

$$\frac{\partial}{\partial W_k}\left\{\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|(W_j - W_i)\right\} = \frac{\partial\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k}(W_j - W_i) + \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|\frac{\partial}{\partial W_k}(W_j - W_i). \tag{4.9}$$

Here the most complicated task is to represent the derivative of $\frac{\partial}{\partial W_k}\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$. Before we go into details we consider the structure of the derivative, that is we consider the mapping

$$\begin{aligned} f:\mathbb{R}^5 &\rightarrow \mathbb{R}^{5\times 5} \\ W &\mapsto \left|\mathbf{A}^{\mathrm{Roe}}\right| = \left(f_{ij}\left(W\right)\right)_{1\le i,j\le 5}, \end{aligned}$$

Its derivative is a linear mapping satisfying

$$df\left[W\right]:\mathbb{R}^5\rightarrow\mathbb{R}^{5\times 5},$$

which can be represented by a matrix $df\left[W\right]\in\mathbb{R}^{5\times(5\times 5)}$, that is we need to compute

$$\begin{aligned} df\left[W\right] &= \left(\frac{\partial f_{ij}}{\partial W}\left[W\right]\right)_{1\le i,j\le 5} \\ &= \left(\frac{\partial f}{\partial\rho},\frac{\partial f}{\partial(\rho u_1)},\frac{\partial f}{\partial(\rho u_2)},\frac{\partial f}{\partial(\rho u_3)},\frac{\partial f}{\partial(\rho E)}\right)\left[W\right]\in\mathbb{R}^{5\times(5\times 5)}. \end{aligned}$$

As a consequence, the term above is understood as

$$\begin{aligned} \frac{\partial\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k}\left(W_j-W_i\right) &= \frac{\partial f}{\partial\rho_k}\left(\rho_j-\rho_i\right)+\frac{\partial f}{\partial\left(\rho u_1\right)_k}\left(\left(\rho u_1\right)_j-\left(\rho u_1\right)_i\right) \\ &+ \frac{\partial f}{\partial\left(\rho u_2\right)_k}\left(\left(\rho u_2\right)_j-\left(\rho u_2\right)_i\right)+\frac{\partial f}{\partial\left(\rho u_3\right)_k}\left(\left(\rho u_3\right)_j-\left(\rho u_3\right)_i\right) \\ &+ \frac{\partial f}{\partial\left(\rho E\right)_k}\left(\left(\rho E\right)_j-\left(\rho E\right)_i\right). \end{aligned}$$

In a first step note that due to Roe averaging (3.19) we have

$$\frac{\partial}{\partial W_k}\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|=0,\qquad k\neq i,k\neq j. \tag{4.10}$$

Second, due to (3.37) one computes

$$\begin{aligned} \frac{\partial\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k} &= \sum_{n=1}^{3}\left(\frac{\partial\left|\lambda_n\right|_{\mathrm{ef},1}}{\partial W_k}g_nq_n+\left|\lambda_n\right|_{\mathrm{ef},1}\frac{\partial g_n}{\partial W_k}q_n+\left|\lambda_n\right|_{\mathrm{ef},1}g_n\frac{\partial q_n}{\partial W_k}\right) \\ &+ \frac{\partial\left|\lambda_4\right|_{\mathrm{ef},1}}{\partial W_k}g_4q_4+\left|\lambda_4\right|_{\mathrm{ef},1}\frac{\partial g_4}{\partial W_k}q_4+\left|\lambda_4\right|_{\mathrm{ef},1}g_4\frac{\partial q_4}{\partial W_k} \\ &+ \frac{\partial\left|\lambda_5\right|_{\mathrm{ef},1}}{\partial W_k}g_5q_5+\left|\lambda_5\right|_{\mathrm{ef},1}\frac{\partial g_5}{\partial W_k}q_5+\left|\lambda_5\right|_{\mathrm{ef},1}g_5\frac{\partial q_5}{\partial W_k}. \end{aligned} \tag{4.11}$$

To compute these required derivatives term by term, we start with the differentiation of the Roe averaged velocity variables (3.19b). We define

$$\begin{aligned} \left(u_{ij,\mathrm{Roe}}\right)_k^{\mathrm{nom}} &:= \left(u_i\right)_k\sqrt{\rho_i}+\left(u_j\right)_k\sqrt{\rho_j}, \\ \left(u_{ij,\mathrm{Roe}}\right)^{\mathrm{dnom}} &:= \sqrt{\rho_i}+\sqrt{\rho_j}, \end{aligned}$$

and compute using the quotient rule

$$\frac{\partial}{\partial W_{i/j}} \left(u_{ij,\mathrm{Roe}}\right)_k = \frac{\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_k^{\mathrm{nom}}}{\partial W_{i/j}} \left(\sqrt{\rho_i} + \sqrt{\rho_j}\right) - \left(u_{ij,\mathrm{Roe}}\right)_k^{\mathrm{nom}} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_k^{\mathrm{dnom}}}{\partial W_{i/j}}}{\left(\sqrt{\rho_i} + \sqrt{\rho_j}\right)^2},$$

where

$$\begin{aligned}
\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1^{\mathrm{nom}}}{\partial W_{i/j}} &= \left(-\frac{u_{i/j}}{2\sqrt{\rho_{i/j}}}, \frac{1}{\sqrt{\rho_{i/j}}}, 0, 0, 0\right), \\
\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2^{\mathrm{nom}}}{\partial W_{i/j}} &= \left(-\frac{u_{i/j}}{2\sqrt{\rho_{i/j}}}, 0, \frac{1}{\sqrt{\rho_{i/j}}}, 0, 0\right), \\
\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3^{\mathrm{nom}}}{\partial W_{i/j}} &= \left(-\frac{u_{i/j}}{2\sqrt{\rho_{i/j}}}, 0, 0, \frac{1}{\sqrt{\rho_{i/j}}}, 0\right),
\end{aligned}$$

and

$$\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)^{\mathrm{dnom}}}{\partial W_{i/j}} = \left(\frac{1}{2\sqrt{\rho_{i/j}}}, 0, 0, 0, 0\right).$$

To find the derivative of $a_{ij,\mathrm{Roe}}$ we exploit its definition (3.19d). So, in a first step we compute

$$\frac{\partial H_{i/j}}{\partial W_{i/j}} = \frac{\partial E_{i/j}}{\partial W_{i/j}} + \frac{\rho_{i/j} \frac{\partial p_{i/j}}{\partial W_{i/j}} - p_{i/j} \frac{\partial \rho_{i/j}}{\partial W_{i/j}}}{\rho_{i/j}^2},$$

where the derivative of pressure p is given in (3.25) and

$$\begin{aligned}
\frac{\partial \rho}{\partial W} &= (1, 0, 0, 0, 0), \\
\frac{\partial E}{\partial W} &= \left(-\frac{E}{\rho}, 0, 0, 0, \frac{1}{\rho}\right).
\end{aligned}$$

Then we define as above

$$\left(H_{ij,\mathrm{Roe}}\right)^{\mathrm{nom}} := H_i \sqrt{\rho_i} + H_j \sqrt{\rho_j}$$

to obtain

$$\frac{\partial H_{ij,\mathrm{Roe}}}{\partial W_{i/j}} = \frac{\frac{\partial \left(H_{ij,\mathrm{Roe}}\right)^{\mathrm{nom}}}{\partial W_{i/j}} \left(\sqrt{\rho_i} + \sqrt{\rho_j}\right) - \left(H_{ij,\mathrm{Roe}}\right)^{\mathrm{nom}} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_k^{\mathrm{dnom}}}{\partial W_{i/j}}}{\left(\sqrt{\rho_i} + \sqrt{\rho_j}\right)^2},$$

where

$$\frac{\partial \left(H_{ij,\mathrm{Roe}}\right)^{\mathrm{nom}}}{\partial W_{i/j}} = H_{i/j} \left(\frac{1}{2\sqrt{\rho_{i/j}}}, 0, 0, 0, 0\right) + \sqrt{\rho_{i/j}} \frac{\partial H_{i/j}}{W_{i/j}}.$$

Then we can compute the derivative of the speed of sound (3.19d) by

$$\begin{aligned}\frac{\partial a_{ij,\mathrm{Roe}}^2}{\partial W_{i/j}} &= (\gamma-1)\left(\frac{\partial H_{ij,\mathrm{Roe}}}{\partial W_{i/j}} - \frac{1}{2}\frac{\partial \left\|u_{ij,\mathrm{Roe}}\right\|_2^2}{\partial W_{i/j}}\right),\\ \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}} &= \frac{1}{2a_{ij,\mathrm{Roe}}}\frac{\partial a_{ij,\mathrm{Roe}}^2}{\partial W_{i/j}},\end{aligned}$$

where the final missing term is given by

$$\frac{\partial \left\|u_{ij,\mathrm{Roe}}\right\|_2^2}{\partial W_{i/j}} = 2\sum_{\ell=1}^{3}\left(u_{ij,\mathrm{Roe}}\right)_\ell \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_\ell}{\partial W_{i/j}}.$$

Using these preparations, to compute $\frac{\partial \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k}$ we start with the derivative of the absolute value of the eigenvalues, that is (3.34), which is given by

$$\frac{\partial \left|V_{ij,\mathrm{Roe}}\right|}{\partial W_{i/j}} = \begin{cases} \frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}}, & V_{ij,\mathrm{Roe}} \geq 0,\\ -\frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}}, & V_{ij,\mathrm{Roe}} < 0.\end{cases} \tag{4.12}$$

where

$$\frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}} = \sum_{\ell=1}^{3} n_{ij,\ell}\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_\ell}{\partial W_{i/j}}.$$

Then we obtain for $\ell = 1,2,3$ the derivative of (3.34)

$$\frac{\partial \left|\lambda_\ell\right|_{\mathrm{ef},1}}{\partial W_{i/j}} = \begin{cases} \frac{\partial \left|V_{ij,\mathrm{Roe}}\right|}{\partial W_{i/j}}, & \left|V_{ij,\mathrm{Roe}}\right| \geq \delta_{\mathrm{ef}}\left(\left|V_{ij,\mathrm{Roe}}\right| + a_{ij,\mathrm{Roe}}\right),\\ \delta_{\mathrm{ef}}\left(\frac{\partial \left|V_{ij,\mathrm{Roe}}\right|}{\partial W_{i/j}} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}}\right), & \left|V_{ij,\mathrm{Roe}}\right| < \delta_{\mathrm{ef}}\left(\left|V_{ij,\mathrm{Roe}}\right| + a_{ij,\mathrm{Roe}}\right),\end{cases},$$

and additionally

$$\frac{\partial \left|\lambda_4\right|_{\mathrm{ef},2}}{\partial W_{i/j}} = \begin{cases} \frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}}, & \begin{array}{c} V_{ij,\mathrm{Roe}} + a_{ij,\mathrm{Roe}} \geq 0 \quad \text{and}\\ \left|V_{ij,\mathrm{Roe}} + a_{ij,\mathrm{Roe}}\right| \geq \delta_{\mathrm{ef}}\left(\left|V_{ij,\mathrm{Roe}}\right| + a_{ij,\mathrm{Roe}}\right)\end{array}\\ -\left(\frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}}\right), & \begin{array}{c} V_{ij,\mathrm{Roe}} + a_{ij,\mathrm{Roe}} < 0 \quad \text{and}\\ \left|V_{ij,\mathrm{Roe}} + a_{ij,\mathrm{Roe}}\right| \geq \delta_{\mathrm{ef}}\left(\left|V_{ij,\mathrm{Roe}}\right| + a_{ij,\mathrm{Roe}}\right)\end{array}\\ \delta_{\mathrm{ef}}\left(\frac{\partial \left|V_{ij,\mathrm{Roe}}\right|}{\partial W_{i/j}} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}}\right), & \left|V_{ij,\mathrm{Roe}} + a_{ij,\mathrm{Roe}}\right| < \delta_{\mathrm{ef}}\left(\left|V_{ij,\mathrm{Roe}}\right| + a_{ij,\mathrm{Roe}}\right),\end{cases}$$

$$\frac{\partial \left|\lambda_5\right|_{\mathrm{ef},3}}{\partial W_{i/j}} = \begin{cases} \frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}} - \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}}, & V_{ij,\mathrm{Roe}} - a_{ij,\mathrm{Roe}} \geq 0 \quad \text{and} \\ & |V_{ij,\mathrm{Roe}} - a_{ij,\mathrm{Roe}}| \geq \delta_{\mathrm{ef}} \left(|V_{ij,\mathrm{Roe}}| + a_{ij,\mathrm{Roe}}\right) \\ -\left(\frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}} - \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}}\right), & V_{ij,\mathrm{Roe}} - a_{ij,\mathrm{Roe}} < 0 \quad \text{and} \\ & |V_{ij,\mathrm{Roe}} - a_{ij,\mathrm{Roe}}| \geq \delta_{\mathrm{ef}} \left(|V_{ij,\mathrm{Roe}}| + a_{ij,\mathrm{Roe}}\right) \\ \delta_{\mathrm{ef}} \left(\frac{\partial \left|V_{ij,\mathrm{Roe}}\right|}{\partial W_{i/j}} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}}\right), & |V_{ij,\mathrm{Roe}} - a_{ij,\mathrm{Roe}}| < \delta_{\mathrm{ef}} \left(|V_{ij,\mathrm{Roe}}| + a_{ij,\mathrm{Roe}}\right). \end{cases}$$

In the next step we compute the derivative of the vectors $g_1, \ldots, g_5$ given in Theorem 3.2.9. These are computed by

$$\begin{aligned} \frac{\partial g_\ell}{\partial W_{i/j}} &= n_\ell \frac{\partial y_1}{\partial W_{i/j}} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}} y_{\ell+1} + a_{ij,\mathrm{Roe}} \frac{\partial y_{\ell+1}}{\partial W_{i/j}}, \qquad \ell = 1, 2, 3, \\ \frac{\partial g_4}{\partial W_{i/j}} &= \frac{\partial a_1}{\partial W_{i/j}} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}} a_2 + a_{ij,\mathrm{Roe}} \frac{\partial a_2}{\partial W_{i/j}}, \\ \frac{\partial g_5}{\partial W_{i/j}} &= \frac{\partial a_1}{\partial W_{i/j}} - \left(\frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}} a_2 + a_{ij,\mathrm{Roe}} \frac{\partial a_2}{\partial W_{i/j}}\right). \end{aligned}$$

For the components we compute

$$\frac{\partial y_1}{\partial W_{i/j}} = \begin{pmatrix} 0 \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} \\ \frac{1}{2} \frac{\partial \left\|u_{ij,\mathrm{Roe}}\right\|_2^2}{\partial W_{i/j}} \end{pmatrix}, \quad \frac{\partial y_2}{\partial W_{i/j}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ n_{ij,3} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} - n_{ij,2} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} \end{pmatrix},$$

$$\frac{\partial y_3}{\partial W_{i/j}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ n_{ij,1} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} - n_{ij,3} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} \end{pmatrix},$$

$$\frac{\partial y_4}{\partial W_{i/j}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ n_{ij,2} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} - n_{ij,1} \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} \end{pmatrix},$$

and

$$\frac{\partial a_1}{\partial W_{i/j}} = \begin{pmatrix} 0 \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} \\ \frac{\partial H_{ij,\mathrm{Roe}}}{\partial W_{i/j}} \end{pmatrix}, \qquad \frac{\partial a_2}{\partial W_{i/j}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}} \end{pmatrix}.$$

Similarly, we obtain for the derivatives of the vectors $q_1, \ldots, q_5$ given in Theorem 3.2.10 the terms

$$\begin{aligned}
\frac{\partial q_\ell}{\partial W_{i/j}} &= n_\ell \frac{\partial r_1^T}{\partial W_{i/j}} - \frac{1}{a_{ij,\mathrm{Roe}}^2} \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}} r_{\ell+1}^t - \frac{1}{a_{ij,\mathrm{Roe}}} \frac{\partial r_{\ell+1}^T}{\partial W_{i/j}}, \qquad \ell = 1, 2, 3, \\
\frac{\partial q_4}{\partial W_{i/j}} &= \frac{1}{2} \left(\frac{-1}{a_{ij,\mathrm{Roe}}^4} \frac{\partial a_{ij,\mathrm{Roe}}^2}{\partial W_{i/j}} b_2^T + \frac{1}{a_{ij,\mathrm{Roe}}^2} \frac{\partial b_2^T}{\partial W_{i/j}} - \frac{1}{a_{ij,\mathrm{Roe}}^2} \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}} b_1^T + \frac{1}{a_{ij,\mathrm{Roe}}} \frac{\partial b_1^T}{\partial W_{i/j}} \right) \\
\frac{\partial q_5}{\partial W_{i/j}} &= \frac{1}{2} \left(\frac{-1}{a_{ij,\mathrm{Roe}}^4} \frac{\partial a_{ij,\mathrm{Roe}}^2}{\partial W_{i/j}} b_2^T + \frac{1}{a_{ij,\mathrm{Roe}}^2} \frac{\partial b_2^T}{\partial W_{i/j}} + \frac{1}{a_{ij,\mathrm{Roe}}^2} \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_{i/j}} b_1^T - \frac{1}{a_{ij,\mathrm{Roe}}} \frac{\partial b_1^T}{\partial W_{i/j}} \right).
\end{aligned}$$

Here the required derivatives are

$$\frac{\partial r_1}{\partial W_{i/j}} = \frac{-1}{a_{ij,\mathrm{Roe}}^4}\frac{\partial a_{ij,\mathrm{Roe}}^2}{\partial W_{i/j}} r_1 + \frac{\gamma-1}{a_{ij,\mathrm{Roe}}^2}\begin{pmatrix} \frac{\partial H_{i/j}}{\partial W_{i/j}} - \frac{\partial \left\|u_{ij,\mathrm{Roe}}\right\|_2^2}{\partial W_{i/j}} \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} \\ \frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} \\ 0 \end{pmatrix},$$

$$\frac{\partial r_2}{\partial W_{i/j}} = \begin{pmatrix} n_{ij,3}\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} - n_{ij,2}\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\frac{\partial r_3}{\partial W_{i/j}} = \begin{pmatrix} n_{ij,1}\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} n_{ij,3}\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\frac{\partial r_4}{\partial W_{i/j}} = \begin{pmatrix} n_{ij,2}\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} - n_{ij,1}\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

and

$$\frac{\partial b_1}{\partial W_{i/j}} = \begin{pmatrix} -\frac{\partial V_{ij,\mathrm{Roe}}}{\partial W_{i/j}} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \frac{\partial b_2}{\partial W_{i/j}} = (\gamma-1)\begin{pmatrix} \frac{1}{2}\frac{\partial \left\|u_{ij,\mathrm{Roe}}\right\|_2^2}{\partial W_{i/j}} \\ -\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_1}{\partial W_{i/j}} \\ -\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_2}{\partial W_{i/j}} \\ -\frac{\partial \left(u_{ij,\mathrm{Roe}}\right)_3}{\partial W_{i/j}} \\ 0 \end{pmatrix}.$$

This concludes the derivative of $\frac{\partial \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k}$, and using (4.11) it can be evaluated.

4.2.2 Derivative of compact inviscid flux and constant $\left|\mathbf{A}^{\mathrm{Roe}}\right|$

The complete derivative $\frac{\partial}{\partial W_k}\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$ has a tremendous operational count. Hence, incorporating this complete derivative into a preconditioner generates possibly an

inefficient algorithm. Considering (4.9) and assuming that the difference $W_j - W_i$ is small, we may neglect in (4.9) the term $\frac{\partial \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k}(W_j - W_i)$. Formally we assume that $\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$ is constant, that is we assume

$$\frac{\partial \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k} = 0.$$

Then we obtain using (4.8) an approximate derivative by

$$\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}(W_i, W_j, n)}{\partial W_k} \approx \frac{1}{2}\begin{cases} \frac{\partial \langle f_c, n\rangle[W_i]}{\partial W_i} + \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|, & k = i, \\ \frac{\partial \langle f_c, n\rangle[W_j]}{\partial W_j} - \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|, & k = j, \\ 0, & k \neq i, j. \end{cases}$$

Hence, the approximate derivative for the corresponding inviscid part of the residual is obtained by

$$\sum_{j\in\mathcal{N}(i)} \frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_k} \approx \frac{1}{2}\begin{cases} \sum_{j\in\mathcal{N}(i)} \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|, & k = i, \\ \frac{\partial \langle f_c, n\rangle[W_k]}{\partial W_k} - \left|\mathbf{A}_{ik}^{\mathrm{Roe}}\right|, & k \in \mathcal{N}(i), \\ 0, & k \neq i, k \notin \mathcal{N}(i). \end{cases} \tag{4.13}$$

Since the terms $\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|$ are required to evaluate the discretization, it is not of additional computational effort to use these terms for the construction of an approximate derivative. This simple fact is exploited to construct a preconditioner to obtain an efficient solution algorithm (see Section 5.2.2).

4.2.3 Derivative of full inviscid flux

To complete the derivative of the inviscid flux, differentiation of the entire numerical flux function $\mathcal{H}$ given in (3.15) in Definition 3.2.3 is required. Naturally, the derivative of the convective flux terms is given by (4.8). To derive the dissipative part, let us write down the term d_{ij} for the case $\mathbf{P}_{ij} = I$ explictly,

$$\begin{aligned} d_{ij}\left(W_{\mathcal{N}(i,j)}, n\right) &= \frac{1}{2}\left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|\left(d_{ij}^{(1)}\left(W_{\mathcal{N}(i,j)}, n\right) - \xi d_{ij}^{(2)}\left(W_{\mathcal{N}(i,j)}, n\right)\right), \\ d_{ij}^{(1)}\left(W_{\mathcal{N}(i,j)}, n\right) &:= \Psi_{ij}\left(W_j - W_i\right), \\ d_{ij}^{(2)}\left(W_{\mathcal{N}(i,j)}, n\right) &:= s_{ij}\left(W_{\mathcal{N}(i,j)}\right)\left(1 - \Psi_{ij}\right)\left(L_j\left(W_{\mathcal{N}(i,j)}\right) - L_i\left(W_{\mathcal{N}(i,j)}\right)\right). \end{aligned}$$

Differentiating gives

$$\frac{\partial d_{ij}}{\partial W_k} = \frac{1}{2}\left\{\left(\frac{\partial \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|}{\partial W_k}\right)\left(d_{ij}^{(1)} - d_{ij}^{(2)}\right) + \left|\mathbf{A}_{ij}^{\mathrm{Roe}}\right|\left(\frac{\partial d_{ij}^{(1)}}{\partial W_k} - \xi\frac{\partial d_{ij}^{(2)}}{\partial W_k}\right)\right\}. \tag{4.14}$$

The first term on the right hand side of (4.14) is completely covered by the considerations in Section 4.2.1. In particular, note that this part of the derivative of d_{ij}

vanishes for $k \notin \{i, \mathcal{N}(i)\}$ due to (4.10). For the second term on the right hand side of (4.14), let us start with

$$\frac{\partial d_{ij}^{(1)}}{\partial W_k} = \begin{cases} \frac{\partial \psi_{ij}}{\partial W_i}(W_j - W_i) - \psi_{ij}, & k = i, \\ \frac{\partial \psi_{ij}}{\partial W_j}(W_j - W_i) + \psi_{ij}, & k = j, \\ 0, & k \notin \{i, \mathcal{N}(i)\}, \end{cases}$$

where

$$\frac{\partial \psi_{ij}}{\partial W_k} = \begin{cases} 0, & \varepsilon_\psi \psi_{ij}^* > 1, \\ \varepsilon_\psi \frac{\partial \psi_{ij}^*}{\partial W_k}, & k \in \{i, j\}, \end{cases}$$

$$\frac{\partial \psi_{ij}^*}{\partial W_k} = \begin{cases} \frac{-2(p_j - p_i)(p_j + p_i)^2 \frac{\partial p_j}{\partial W_i} - 2(p_j - p_i)^2 (p_j + p_i) \frac{\partial p_j}{\partial W_i}}{(p_j + p_i)^4}, & k = i, \\ \frac{2(p_j - p_i)(p_j + p_i)^2 \frac{\partial p_j}{\partial W_j} - 2(p_j - p_i)^2 (p_j + p_i) \frac{\partial p_j}{\partial W_j}}{(p_j + p_i)^4}, & k = j. \end{cases}$$

The derivative for pressure p can be found in (3.25). The derivative for $d_{ij}^{(2)}$ is computed by

$$\begin{aligned} \frac{\partial d_{ij}^{(2)}}{\partial W_k} &= (1 - \Psi_{ij})(L_j - L_i) \frac{\partial s_{ij}}{\partial W_k} + s_{ij}(L_j - L_i) \frac{\partial (1 - \Psi_{ij})}{\partial W_k} \\ &\quad + s_{ij}(1 - \Psi_{ij}) \frac{\partial (L_j - L_i)}{\partial W_k}. \end{aligned}$$

The derivative for

$$\frac{\partial (1 - \Psi_{ij})}{\partial W_k} = -\frac{\partial \Psi_{ij}}{\partial W_k}$$

is already computed above, and

$$\frac{\partial (L_j - L_i)}{\partial W_k} = \frac{\partial (L_j - L_i)}{\partial (L_i, L_j)} \begin{pmatrix} \frac{\partial L_i}{\partial W_k} \\ \frac{\partial L_j}{\partial W_k} \end{pmatrix} = (-1, 1) \begin{pmatrix} \frac{\partial L_i}{\partial W_k} \\ \frac{\partial L_j}{\partial W_k} \end{pmatrix} = \frac{\partial L_j}{\partial W_k} - \frac{\partial L_i}{\partial W_k}$$

was already considered in Section 4.1.2. The final missing term is also computed using the chain rule,

$$\frac{\partial s_{ij}}{\partial W_k} = \frac{\partial s_{ij}}{\partial (z_i, z_j)} \begin{pmatrix} \frac{\partial z_i}{\partial W_k} \\ \frac{\partial z_j}{\partial W_k} \end{pmatrix} = \frac{\partial s_{ij}}{\partial (z_i, z_j)} \begin{pmatrix} \frac{\partial z_i}{\partial (\lambda_{i,\mathrm{Roe}}, \lambda_{ij,\mathrm{Roe}})} \begin{pmatrix} \frac{\partial \lambda_{i,\mathrm{Roe}}}{\partial W_k} \\ \frac{\partial \lambda_{ij,\mathrm{Roe}}}{\partial W_k} \end{pmatrix} \\ \frac{\partial z_j}{\partial (\lambda_{j,\mathrm{Roe}}, \lambda_{ij,\mathrm{Roe}})} \begin{pmatrix} \frac{\partial \lambda_{j,\mathrm{Roe}}}{\partial W_k} \\ \frac{\partial \lambda_{ij,\mathrm{Roe}}}{\partial W_k} \end{pmatrix} \end{pmatrix}.$$

Using (4.12) the derivative $\frac{\partial \lambda_{ij,\mathrm{Roe}}}{\partial W_k}$ is given by

$$\frac{\partial \lambda_{ij,\mathrm{Roe}}}{\partial W_k} = \begin{cases} \frac{\partial |V_{ij,\mathrm{Roe}}|}{\partial W_i} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_i}, & k = i, \\ \frac{\partial |V_{ij,\mathrm{Roe}}|}{\partial W_j} + \frac{\partial a_{ij,\mathrm{Roe}}}{\partial W_j}, & k = j, \\ 0, & k \neq i, k \neq j, \end{cases}$$

and hence

$$\frac{\partial \lambda_{i,\mathrm{Roe}}}{\partial W_k} = \begin{cases} \sum_{j \in \mathcal{N}(i)} \frac{\partial \lambda_{ij,\mathrm{Roe}}}{\partial W_i}, & k = i, \\ \frac{\partial \lambda_{ij,\mathrm{Roe}}}{\partial W_j}. & k = j, \\ 0, & k \neq i, k \neq j. \end{cases}$$

Furthermore, we obtain

$$\begin{aligned} \frac{\partial z_i}{\partial \left(\lambda_{i,\mathrm{Roe}}, \lambda_{ij,\mathrm{Roe}}\right)} &= \left(\frac{1}{\mathrm{svol}(e_{ij}) \lambda_{ij,\mathrm{Roe}}}, -\frac{\lambda_{i,\mathrm{Roe}}}{\mathrm{svol}(e_{ij})} \frac{1}{\lambda_{ij,\mathrm{Roe}}^2} \right), \\ \frac{\partial s_{ij}}{\partial (z_i, z_j)} &= \left(\frac{z_j}{\sqrt{z_i}\left(z_i + 2\sqrt{z_i z_j} + z_j\right)}, \frac{z_i}{\sqrt{z_j}\left(z_i + 2\sqrt{z_i z_j} + z_j\right)} \right). \end{aligned}$$

This completes the construction of the full derivative of inviscid terms. According to the considerations in Section 4.1.2 several of these terms need to be pre-computed, such that the chain rule can be applied within an algorithmical structure.

4.3 Derivative of viscous terms

The goal of this section is to present the derivative of the $\left\langle f_v\left(W_{\mathcal{N}(i,j)}\right), n_{e_{ij}}\right\rangle$ defined in (3.71). To shorten the notation, throughout this section we neglect the edge (face) identifier e_{ij} for the normal, that is we write $n_{e_{ij}} = n = (n_1, n_2, n_3)^T$.

Similar as for the inviscid terms a differentiation of the the actual discretization is required. In general the part of residual $\mathbf{R}_i$ corresponding to $\left\langle f_v\left(W_{\mathcal{N}(i,j)}\right), n_{e_{ij}}\right\rangle$ is a term which depends on the variables corresponding to i, $\mathcal{N}(i)$ and $\mathcal{N}\left(\mathcal{N}(i)\right)$. Restricting ourselves to approximate gradients using the thin layer assumption (3.65) only, the corresponding residual $\mathbf{R}_i$ is compact. For simplicity of presentation in this thesis we restrict ourselves to this case. The more general case where the gradients are computed by a Green-Gauss method (3.60) and (3.63) can be realized similar to explanations in Section 4.1.1, by differentiation and application of the chain rule.

4.3.1 Derivative of viscous terms assuming TSL

Using a TSL the viscous flux on the face e_{ij} depends only on the variables W_i and W_j. Hence, we need to compute

$$\frac{\partial \left\langle f_v(W_i, W_j), n\right\rangle}{\partial W_i} \qquad \text{and} \qquad \frac{\partial \left\langle f_v(W_i, W_j), n\right\rangle}{\partial W_j},$$

since all other terms vanish,

$$\frac{\partial \left\langle f_v(W_i, W_j), n\right\rangle}{\partial W_k} = 0, \qquad k \neq i, \quad k \neq j.$$

To compute these derivatives we exploit the equation of state (2.3) and the derivative (3.25) of the pressure to obtain

$$\frac{\partial T}{\partial W} = \frac{\partial\left(\frac{p}{\rho}\right)}{\partial W} = \frac{\rho\frac{\partial p}{\partial W} - p\frac{\partial \rho}{\partial W}}{\rho^2} = \frac{1}{\rho}\left(\frac{(\gamma-1)}{2}\|u\|_2^2 - T, (1-\gamma)u, (\gamma-1)\right).$$

Furthermore, the derivative of velocity is determined as

$$\frac{\partial u}{\partial W} = \frac{1}{\rho}\begin{pmatrix} -u_1 & 1 & 0 & 0 & 0 \\ -u_2 & 0 & 1 & 0 & 0 \\ -u_3 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.15}$$

Therefore, we get for the discrete TSL gradient of the velocity and temperature

$$\left(\frac{\partial u_\ell}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = n_{k,ij}\frac{(u_{j,\ell} - u_{i,\ell})}{\mathrm{dist}(e_{ij})}, \qquad \ell = 1,2,3, \tag{4.16}$$

$$\left(\frac{\partial u}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = \left(\left(\frac{\partial u_1}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}}, \left(\frac{\partial u_2}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}}, \left(\frac{\partial u_3}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}}\right)^T, \tag{4.17}$$

$$\left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = n_{k,ij}\frac{(T_j - T_i)}{\mathrm{dist}(e_{ij})}, \tag{4.18}$$

the derivatives

$$\frac{\partial}{\partial W_i}\left(\frac{\partial u}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_i}\begin{pmatrix} u_{i,1} & -1 & 0 & 0 & 0 \\ u_{i,2} & 0 & -1 & 0 & 0 \\ u_{i,3} & 0 & 0 & -1 & 0 \end{pmatrix}, \tag{4.19}$$

$$\frac{\partial}{\partial W_j}\left(\frac{\partial u}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\begin{pmatrix} -u_{j,1} & 1 & 0 & 0 & 0 \\ -u_{j,2} & 0 & 1 & 0 & 0 \\ -u_{j,3} & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{4.20}$$

$$\frac{\partial}{\partial W_i}\left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_i}\left(T_i - \frac{(\gamma-1)}{2}\|u_i\|_2^2, (\gamma-1)u_i, 1-\gamma\right), \tag{4.21}$$

$$\frac{\partial}{\partial W_j}\left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} = \frac{n_{k,ij}}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(\frac{(\gamma-1)}{2}\|u_j\|_2^2 - T_j, (1-\gamma)u_j, \gamma-1\right). \tag{4.22}$$

We use these preparations to compute the derivative of the the trace free shear stress tensor $\overline{\mathcal{S}}$ given in Definition 2.1.1.

Lemma 4.3.1 *Using a* TSL *approximation for the gradients, the derivative of the trace free shear stress tensor* $\overline{\mathcal{S}}_{e_{ij}} = \overline{\mathcal{S}}_{e_{ij}}^{\mathrm{TSL}}$ *in normal direction* $n = (n_1, n_2, n_3)^T$ *is given by*

$$\frac{\partial\left\langle\left(\overline{\mathcal{S}}\right)_{e_{ij}}^{\mathrm{TSL}}, n\right\rangle}{\partial W_j} = \xi^{(j)}\begin{pmatrix} -N_1 u_{j,1} - N_{12,13}^{(j)} & N_1 & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} & 0 \\ -N_2 u_{j,2} - N_{12,23}^{(j)} & \frac{1}{3}N_{12} & N_2 & \frac{1}{3}N_{23} & 0 \\ -N_3 u_{j,3} - N_{13,23}^{(j)} & \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & N_3 & 0 \end{pmatrix}, \tag{4.23}$$

and

$$\frac{\partial \left\langle \left(\overline{\mathcal{S}}\right)_{e_{ij}}^{\mathrm{TSL}}, n \right\rangle}{\partial W_i} = \xi^{(i)} \begin{pmatrix} N_1 u_{i,1} + N_{12,13}^{(i)} & -N_1 & -\frac{1}{3}N_{12} & -\frac{1}{3}N_{13} & 0 \\ N_2 u_{i,2} + N_{12,23}^{(i)} & -\frac{1}{3}N_{12} & -N_2 & -\frac{1}{3}N_{23} & 0 \\ N_3 u_{i,3} + N_{13,23}^{(i)} & -\frac{1}{3}N_{13} & -\frac{1}{3}N_{23} & -N_3 & 0 \end{pmatrix}, \quad (4.24)$$

where

$$\begin{aligned} \xi^{(k)} &:= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_k}, \qquad N_1 := \frac{4}{3}n_1^2 + n_2^2 + n_3^2, \\ N_2 &:= n_1^2 + \frac{4}{3}n_2^2 + n_3^2, \qquad N_3 := n_1^2 + n_2^2 + \frac{4}{3}n_3^2, \\ N_{12} &:= n_1 n_2, \qquad N_{13} := n_1 n_3, \qquad N_{23} := n_2 n_3, \\ N_{12,13}^{(k)} &:= \frac{1}{3}\left(N_{12}u_{k,2} + N_{13}u_{k,3}\right), \qquad N_{12,23}^{(k)} := \frac{1}{3}\left(N_{12}u_{k,1} + N_{23}u_{k,3}\right), \\ N_{13,23}^{(k)} &:= \frac{1}{3}\left(N_{13}u_{k,1} + N_{23}u_{k,3}\right), \qquad k = i, j. \end{aligned}$$

Proof: Using the definition (2.6), in discretized form the trace free shear stress tensor $\overline{\mathcal{S}}_{e_{ij}} = \overline{\mathcal{S}}_{e_{ij}}^{\mathrm{TSL}}$ reads

$$\begin{aligned} \left(\overline{\mathcal{S}}_{\ell\ell}\right)_{e_{ij}}^{\mathrm{TSL}} &= \frac{1}{3}\left(2\left(\frac{\partial u_\ell}{\partial x_\ell}\right)_{e_{ij}}^{\mathrm{TSL}} - \sum_{k=1,k\neq\ell}^{m}\left(\frac{\partial u_k}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}}\right), \\ \left(\overline{\mathcal{S}}_{\ell k}\right)_{e_{ij}}^{\mathrm{TSL}} &= \frac{1}{2}\left(\left(\frac{\partial u_\ell}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}} + \left(\frac{\partial u_k}{\partial x_\ell}\right)_{e_{ij}}^{\mathrm{TSL}}\right), \qquad k \neq l. \end{aligned}$$

Then by an application of (4.16), (4.19) and (4.20) we obtain

$$\begin{aligned} \frac{\partial \left(\overline{\mathcal{S}}_{11}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \rho_j} &= \frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(-2n_1u_{j,1} + n_2u_{j,2} + n_3u_{j,3}\right), \\ \frac{\partial \left(\overline{\mathcal{S}}_{11}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_1)_j} &= \frac{2}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_1}{\rho_j}, \\ \frac{\partial \left(\overline{\mathcal{S}}_{11}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_2)_j} &= -\frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_2}{\rho_j}, \\ \frac{\partial \left(\overline{\mathcal{S}}_{11}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_3)_j} &= -\frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_3}{\rho_j}, \\ \frac{\partial \left(\overline{\mathcal{S}}_{11}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho E)_j} &= 0, \end{aligned}$$

$$
\begin{aligned}
\frac{\partial \left(\overline{\mathcal{S}}_{22}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \rho_j} &= \frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(-2n_2u_{j,2}+n_1u_{j,1}+n_3u_{j,3}\right),\\
\frac{\partial \left(\overline{\mathcal{S}}_{22}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_1)_j} &= -\frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_1}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{22}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_2)_j} &= \frac{2}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_2}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{22}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_3)_j} &= -\frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_3}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{22}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho E)_j} &= 0,
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \left(\overline{\mathcal{S}}_{33}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \rho_j} &= \frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(-2n_3u_{j,3}+n_1u_{j,1}+n_2u_{j,2}\right),\\
\frac{\partial \left(\overline{\mathcal{S}}_{33}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_1)_j} &= -\frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_1}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{33}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_2)_j} &= -\frac{1}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_2}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{33}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_3)_j} &= \frac{2}{3}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_3}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{33}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho E)_j} &= 0.
\end{aligned}
$$

The cross derivatives are given by

$$
\begin{aligned}
\frac{\partial \left(\overline{\mathcal{S}}_{12}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \rho_j} &= -\frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(n_2u_{j,1}+n_1u_{j,2}\right),\\
\frac{\partial \left(\overline{\mathcal{S}}_{12}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_1)_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_2}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{12}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_2)_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_1}{\rho_j},\\
\frac{\partial \left(\overline{\mathcal{S}}_{12}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_3)_j} &= \frac{\partial \overline{\mathcal{S}}_{12}}{\partial (\rho E)_j} = 0.
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \left(\overline{\mathcal{S}}_{13}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \rho_j} &= -\frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(n_3 u_{j,1} + n_1 u_{j,3}\right), \\
\frac{\partial \left(\overline{\mathcal{S}}_{13}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_1)_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_3}{\rho_j}, \\
\frac{\partial \left(\overline{\mathcal{S}}_{13}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_2)_j} &= \frac{\partial \overline{\mathcal{S}}_{13}}{\partial (\rho E)_j} = 0, \\
\frac{\partial \left(\overline{\mathcal{S}}_{13}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_3)_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_1}{\rho_j},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \left(\overline{\mathcal{S}}_{23}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \rho_j} &= -\frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(n_3 u_{j,2} + n_2 u_{j,3}\right), \\
\frac{\partial \left(\overline{\mathcal{S}}_{23}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_1)_j} &= \frac{\partial \overline{\mathcal{S}}_{23}}{\partial (\rho E)_j} = 0, \\
\frac{\partial \left(\overline{\mathcal{S}}_{23}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_2)_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_3}{\rho_j}, \\
\frac{\partial \left(\overline{\mathcal{S}}_{23}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial (\rho u_3)_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{n_2}{\rho_j}.
\end{aligned}
$$

The remainder cross derivatives are a consequence of the symmetries $\left(\overline{\mathcal{S}}_{e_{ij}}\right)^{\mathrm{TSL}} = \left(\overline{\mathcal{S}}_{e_{ji}}\right)^{\mathrm{TSL}}$, $i \neq j$. In summary, we compute

$$
\begin{aligned}
\frac{\partial \left\langle \left(\overline{\mathcal{S}}_1\right)_{e_{ij}}^{\mathrm{TSL}}, n_{e_{ij}} \right\rangle}{\partial W_j} &= \sum_{k=1}^{3} n_k \frac{\partial \left(\overline{\mathcal{S}}_{1k}\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial W_j} \\
&= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(-N_1 u_{j,1} - N_{12,13}^{(j)}, N_1, \frac{1}{3}N_{12}, \frac{1}{3}N_{13}, 0\right).
\end{aligned}
$$

Evaluating the other components straightforward we obtain

$$
\begin{aligned}
\frac{\partial \left\langle \left(\overline{\mathcal{S}}_2\right)_{e_{ij}}^{\mathrm{TSL}}, n_{e_{ij}} \right\rangle}{\partial W_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(-N_2 u_{j,2} - N_{12,23}^{(j)}, \frac{1}{3}N_{12}, N_2, \frac{1}{3}N_{23}, 0\right), \\
\frac{\partial \left\langle \left(\overline{\mathcal{S}}_3\right)_{e_{ij}}^{\mathrm{TSL}}, n_{e_{ij}} \right\rangle}{\partial W_j} &= \frac{1}{2}\frac{1}{\mathrm{dist}(e_{ij})}\frac{1}{\rho_j}\left(-N_3 u_{j,3} - N_{13,23}^{(j)}, \frac{1}{3}N_{13}, \frac{1}{3}N_{23}, N_3, 0\right).
\end{aligned}
$$

Equation (4.24) follows line by line by the computations above.

□

Multiplying $\overline{\mathcal{S}}^{\mathrm{TSL}}_{e_{ij}}$ by the averaged viscosity (3.66) we obtain for $m = i, j$ as derivative for the viscous stress tensor

$$\begin{aligned}\frac{\partial \left\langle (\tau)^{\mathrm{TSL}}_{e_{ij}}, n \right\rangle}{\partial W_m} &= \frac{\partial \left\langle \left(2\mu_{\mathrm{eff},e_{ij}} \overline{\mathcal{S}}\right)^{\mathrm{TSL}}_{e_{ij}}, n \right\rangle}{\partial W_m} \\ &= 2\left[\frac{\partial \mu_{\mathrm{eff},e_{ij}}}{\partial W_m}\left\langle \left(\overline{\mathcal{S}}\right)^{\mathrm{TSL}}_{e_{ij}}, n \right\rangle + \mu_{\mathrm{eff},e_{ij}} \frac{\partial \left\langle \left(\overline{\mathcal{S}}\right)^{\mathrm{TSL}}_{e_{ij}}, n \right\rangle}{\partial W_m}\right], \end{aligned} \tag{4.25}$$

and the derivative of Sutherland's law can be computed as

$$\begin{aligned}\frac{\partial \mu_l}{\partial W} &= \frac{\partial \mu_l}{\partial T}\frac{\partial T}{\partial W} \\ &= \mu_{l,\infty}\left\{\frac{3}{2}\sqrt{T}\left(\frac{1+C_{\mathrm{suth}}}{T+C_{\mathrm{suth}}}\right) - T^{3/2}\left(\frac{1+C_{\mathrm{suth}}}{(T+C_{\mathrm{suth}})^2}\right)\right\}\frac{\partial T}{\partial W} \\ &= \mu_{l,\infty}\left\{\sqrt{T}\left(\frac{1+C_{\mathrm{suth}}}{T+C_{\mathrm{suth}}}\right)\left[\frac{3}{2} - T\left(\frac{1}{T+C_{\mathrm{suth}}}\right)\right]\right\}\frac{\partial T}{\partial W} \\ &= \mu_{l,\infty}\left\{\frac{\sqrt{T}}{2}\frac{(1+C_{\mathrm{suth}})(T+3C_{\mathrm{suth}})}{(T+C_{\mathrm{suth}})^2}\right\}\frac{\partial T}{\partial W}. \end{aligned} \tag{4.26}$$

The derivate of the eddy viscosity μ_t depends on the turbulence model and cannot be stated in general. Finally, we compute for the remaining component of the viscous flux $\left(\langle f_v\left(W_i, W_j\right), n\rangle\right)_5 = \langle \theta, n\rangle$, using the representation (2.8), the derivative

$$\frac{\partial \left\langle (\theta)^{\mathrm{TSL}}_{e_{ij}}, n \right\rangle}{\partial W_m} = \frac{\partial \left\langle (\tau)^{\mathrm{TSL}}_{e_{ij}} u_{e_{ij}}, n \right\rangle}{\partial W_m} + \frac{\partial \left\langle (q)^{\mathrm{TSL}}_{e_{ij}}, n \right\rangle}{\partial W_m}. \tag{4.27}$$

For the first term on the right hand side of (4.27) we compute for $m = i, j$

$$\begin{aligned}&\frac{\partial \left\langle (\tau)^{\mathrm{TSL}}_{e_{ij}} u_{e_{ij}}, n \right\rangle}{\partial W_m} \\ &= \left(\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} \frac{\partial}{\partial W_m}\left(u_{\ell,e_{ij}} \left(\tau_{k\ell}\right)^{\mathrm{TSL}}_{e_{ij}}\right)\right) \\ &= \left(\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} \left(\tau_{k\ell}\right)^{\mathrm{TSL}}_{e_{ij}} \frac{\partial u_{\ell,e_{ij}}}{\partial W_m}\right) + \left(\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} u_{\ell,e_{ij}} \frac{\partial \left(\tau_{k\ell}\right)^{\mathrm{TSL}}_{e_{ij}}}{\partial W_m}\right) \\ &= \left(\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} \left(\tau_{k\ell}\right)^{\mathrm{TSL}}_{e_{ij}} \frac{\partial u_{\ell,e_{ij}}}{\partial W_m}\right) + \left(\sum_{\ell=1}^{3} u_{\ell,e_{ij}} \sum_{k=1}^{3} n_k \frac{\partial \left(\tau_{k\ell}\right)^{\mathrm{TSL}}_{e_{ij}}}{\partial W_m}\right) \\ &= \left(\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} \left(\tau_{k\ell}\right)^{\mathrm{TSL}}_{e_{ij}} \frac{\partial u_{\ell,e_{ij}}}{\partial W_m}\right) + \left(\sum_{\ell=1}^{3} u_{\ell,e_{ij}} \left\langle \frac{\partial \left(\tau_\ell\right)^{\mathrm{TSL}}_{e_{ij}}}{\partial W_m}, n \right\rangle\right). \end{aligned} \tag{4.28}$$

Using (3.68) and (4.15) we have

$$\frac{\partial u_{e_{ij}}}{\partial W_m} = \frac{1}{2}\frac{1}{\rho_m}\begin{pmatrix} -u_{m,1} & 1 & 0 & 0 & 0 \\ -u_{m,2} & 0 & 1 & 0 & 0 \\ -u_{m,3} & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad m = i, j. \tag{4.29}$$

Equation (4.29) is used to compute explicitly the first term on the right hand side of (4.28). We obtain

$$\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} \left(\tau_{k\ell}\right)_{e_{ij}}^{\mathrm{TSL}} \frac{\partial u_{\ell,e_{ij}}}{\partial \rho_m} = -\frac{1}{2}\frac{1}{\rho_m}\sum_{\ell=1}^{3} u_{m,\ell} \sum_{k=1}^{3} n_k \left(\tau_{k\ell}\right)_{e_{ij}}^{\mathrm{TSL}}, \tag{4.30}$$

$$\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} \left(\tau_{k\ell}\right)_{e_{ij}}^{\mathrm{TSL}} \frac{\partial u_{\ell,e_{ij}}}{\partial \left(\rho u\right)_m} = \frac{1}{2}\frac{1}{\rho_m}\begin{pmatrix} \langle n, \tau_1 \rangle \\ \langle n, \tau_2 \rangle \\ \langle n, \tau_3 \rangle \end{pmatrix}^T, \tag{4.31}$$

$$\sum_{k=1}^{3} n_k \sum_{\ell=1}^{3} \left(\tau_{k\ell}\right)_{e_{ij}}^{\mathrm{TSL}} \frac{\partial u_{\ell,e_{ij}}}{\partial \left(\rho E\right)_m} = 0. \tag{4.32}$$

For the second term on the right hand side of (4.27) we compute

$$\begin{aligned} & \frac{\partial \left\langle \left(q\right)_{e_{ij}}^{\mathrm{TSL}}, n \right\rangle}{\partial W_m} \\ = \; & \frac{\partial}{\partial W_m}\left[\kappa_{\mathrm{eff},e_{ij}} \left(\sum_{k=1}^{3} n_k \left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}}\right)\right] \\ = \; & \frac{\partial \kappa_{\mathrm{eff},e_{ij}}}{\partial W_m}\left(\sum_{k=1}^{3} n_k \left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}}\right) + \kappa_{\mathrm{eff},e_{ij}}\left(\sum_{k=1}^{3} n_k \frac{\partial}{\partial W_m}\left(\frac{\partial T}{\partial x_k}\right)_{e_{ij}}^{\mathrm{TSL}}\right). \end{aligned} \tag{4.33}$$

Due to (2.11) and (2.14) the derivative of κ_l is directly related to (4.26) and the derivative of κ_t to the eddy viscosity μ_t.

Corollary 4.3.2 *Equations (4.23 (4.25), (4.26), (4.27), (4.28) and (4.33) represent the derivative of $\left\langle f_v, n_{e_{ij}} \right\rangle$ assuming a TSL approximation of the velocity and temperature gradients.*

We close this section considering a further important simplification. Equations (4.25) and (4.33) above show that assuming both constant laminar viscosity and eddy viscosity several terms in the derivative vanish and we can represent the derivatives explicitly.

Corollary 4.3.3 *Assuming a TSL approximation of velocity and temperature gradients and that the viscosity μ_{eff} is constant, then we have*

$$\left(\frac{\partial \left\langle f_v\left(W_i, W_j\right), n\right\rangle}{\partial W_j}\right)^{\text{TSL},\mu=\text{const}} = \frac{\mu_{\text{eff},e_{ij}}}{\text{dist}(e_{ij})}\frac{1}{\rho_j}$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -N_1u_{j,1} - N_{12,13}^{(j)} & N_1 & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} & 0 \\ -N_2u_{j,2} - N_{12,23}^{(j)} & \frac{1}{3}N_{12} & N_2 & \frac{1}{3}N_{23} & 0 \\ -N_3u_{j,3} - N_{13,23}^{(j)} & \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & N_3 & 0 \\ -E_1^{(j)} + E_2^{(j)} + E_3^{(j)} & E_4^{(j)} & E_5^{(j)} & E_6^{(j)} & \frac{\kappa_{\text{eff},e_{ij}}(\gamma-1)}{\mu_{\text{eff},e_{ij}}} \end{pmatrix}, \quad (4.34)$$

and

$$\left(\frac{\partial \left\langle f_v\left(W_i, W_j\right), n\right\rangle}{\partial W_i}\right)^{\text{TSL},\mu=\text{const}} = \frac{\mu_{\text{eff},e_{ij}}}{\text{dist}(e_{ij})}\frac{1}{\rho_i}$$
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ N_1u_{i,1} + N_{12,13}^{(i)} & -N_1 & -\frac{1}{3}N_{12} & -\frac{1}{3}N_{13} & 0 \\ N_2u_{i,2} + N_{12,23}^{(i)} & -\frac{1}{3}N_{12} & -N_2 & -\frac{1}{3}N_{23} & 0 \\ N_3u_{i,3} + N_{13,23}^{(i)} & -\frac{1}{3}N_{13} & -\frac{1}{3}N_{23} & -N_3 & 0 \\ E_1^{(i)} + E_2^{(i)} + E_3^{(i)} & E_4^{(i)} & E_5^{(i)} & E_6^{(i)} & \frac{\kappa_{\text{eff},e_{ij}}(1-\gamma)}{\mu_{\text{eff},e_{ij}}} \end{pmatrix}. \quad (4.35)$$

Here we have used the notation of Lemma 4.3.1 and

$$\begin{aligned} E_1^{(j)} &= u_{1,e_{ij}}\left(N_1u_{j,1} + N_{12,13}^{(j)}\right) + u_{2,e_{ij}}\left(N_2u_{j,2} + N_{12,23}^{(j)}\right) + u_{3,e_{ij}}\left(N_3u_{j,3} + N_{13,23}^{(j)}\right), \\ E_1^{(i)} &= u_{1,e_{ij}}\left(N_1u_{i,1} + N_{12,13}^{(i)}\right) + u_{2,e_{ij}}\left(N_2u_{i,2} + N_{12,23}^{(i)}\right) + u_{3,e_{ij}}\left(N_3u_{i,3} + N_{13,23}^{(i)}\right), \end{aligned}$$

and

$$\begin{aligned} E_2^{(j)} &= -\frac{1}{2}\frac{\text{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\sum_{\ell=1}^{3} u_{j,\ell}\sum_{k=1}^{3} n_k\left(\tau_{k\ell}\right)_{e_{ij}}^{\text{TSL}}, \\ E_2^{(i)} &= -\frac{1}{2}\frac{\text{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\sum_{\ell=1}^{3} u_{i,\ell}\sum_{k=1}^{3} n_k\left(\tau_{k\ell}\right)_{e_{ij}}^{\text{TSL}}, \end{aligned}$$

and

$$\begin{aligned} E_3^{(j)} &= \frac{\kappa_{\text{eff},e_{ij}}}{\mu_{\text{eff},e_{ij}}}\left(\frac{\gamma-1}{2}\|u_j\|_2^2 - T_j\right), \\ E_3^{(i)} &= \frac{\kappa_{\text{eff},e_{ij}}}{\mu_{\text{eff},e_{ij}}}\left(T_i - \frac{\gamma-1}{2}\|u_i\|_2^2\right), \end{aligned}$$

as well as

$$
\begin{aligned}
E_4^{(j)} &= \frac{1}{2}\frac{\operatorname{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\langle n,\tau_1\rangle + \left(N_1u_{1,e_{ij}} + \frac{1}{3}N_{12}u_{2,e_{ij}} + \frac{1}{3}N_{13}u_{3,e_{ij}}\right) - \zeta_{e_{ij}}u_{j,1}\\
E_4^{(i)} &= \frac{1}{2}\frac{\operatorname{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\langle n,\tau_1\rangle - \left(N_1u_{1,e_{ij}} + \frac{1}{3}N_{12}u_{2,e_{ij}} + \frac{1}{3}N_{13}u_{3,e_{ij}}\right) + \zeta_{e_{ij}}u_{i,1}\\
E_5^{(j)} &= \frac{1}{2}\frac{\operatorname{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\langle n,\tau_2\rangle + \left(\frac{1}{3}N_{12}u_{1,e_{ij}} + N_2u_{2,e_{ij}} + \frac{1}{3}N_{23}u_{3,e_{ij}}\right) - \zeta_{e_{ij}}u_{j,2}\\
E_5^{(i)} &= \frac{1}{2}\frac{\operatorname{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\langle n,\tau_2\rangle - \left(\frac{1}{3}N_{12}u_{1,e_{ij}} + N_2u_{2,e_{ij}} + \frac{1}{3}N_{23}u_{3,e_{ij}}\right) + \zeta_{e_{ij}}u_{i,2}\\
E_6^{(j)} &= \frac{1}{2}\frac{\operatorname{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\langle n,\tau_3\rangle + \left(\frac{1}{3}N_{13}u_{1,e_{ij}} + \frac{1}{3}N_{23}u_{2,e_{ij}} + N_3u_{3,e_{ij}}\right) - \zeta_{e_{ij}}u_{j,3}\\
E_6^{(i)} &= \frac{1}{2}\frac{\operatorname{dist}(e_{ij})}{\mu_{\text{eff},e_{ij}}}\langle n,\tau_3\rangle - \left(\frac{1}{3}N_{13}u_{1,e_{ij}} + \frac{1}{3}N_{23}u_{2,e_{ij}} + N_3u_{3,e_{ij}}\right) + \zeta_{e_{ij}}u_{i,3},
\end{aligned}
$$

where

$$\zeta_{e_{ij}} := \frac{\kappa_{\text{eff},e_{ij}}}{\mu_{\text{eff},e_{ij}}}(\gamma-1).$$

Proof: The first row in the matrix of (4.34) is a consequence of the definition of f_v and the rows $2,3$ and 4 follow from (4.25) and Lemma 4.3.1. An application of (4.27), (4.28) together with (4.32) and (4.33) together with (4.23) and (4.22) gives

$$\frac{\partial\left\langle(\theta)_{e_{ij}}^{\text{TSL}},n\right\rangle}{\partial(\rho E)_j} = \frac{\kappa_{\text{eff},e_{ij}}}{\operatorname{dist}(e_{ij})}\frac{(\gamma-1)}{\rho_j}\sum_{k=1}^{3}n_k^2 = \frac{\kappa_{\text{eff},e_{ij}}}{\operatorname{dist}(e_{ij})}\frac{(\gamma-1)}{\rho_i},$$

and together with (4.24) and (4.21)

$$\frac{\partial\left\langle(\theta)_{e_{ij}}^{\text{TSL}},n\right\rangle}{\partial(\rho E)_i} = \frac{\kappa_{\text{eff},e_{ij}}}{\operatorname{dist}(e_{ij})}\frac{(1-\gamma)}{\rho_j}\sum_{k=1}^{3}n_k^2 = \frac{\kappa_{\text{eff},e_{ij}}}{\operatorname{dist}(e_{ij})}\frac{(1-\gamma)}{\rho_i}.$$

which proves the last entry in the 5th row of (4.34) and (4.35).

For the first entry in the 5th row we evaluate in a first step using (4.23) and (4.24)

$$
\begin{aligned}
\left(\sum_{\ell=1}^{3}u_{\ell,e_{ij}}\left\langle\frac{\partial(\tau_\ell)_{e_{ij}}^{\text{TSL}}}{\partial\rho_j},n\right\rangle\right) &= -\frac{\mu_{\text{eff},e_{ij}}}{\operatorname{dist}(e_{ij})}\frac{1}{\rho_j}E_1^{(j)}, &(4.36)\\
\left(\sum_{\ell=1}^{3}u_{\ell,e_{ij}}\left\langle\frac{\partial(\tau_\ell)_{e_{ij}}^{\text{TSL}}}{\partial\rho_i},n\right\rangle\right) &= \frac{\mu_{\text{eff},e_{ij}}}{\operatorname{dist}(e_{ij})}\frac{1}{\rho_i}E_1^{(i)}. &(4.37)
\end{aligned}
$$

The representation of $E_2^{(j)}$ and $E_2^{(i)}$ are a consequence of (4.30), and the representation of $E_3^{(j)}$ and $E_3^{(i)}$ of (4.33) together with (4.22) and (4.21). The terms $E_4^{(m)}, E_5^{(m)}$ and $E_6^{(m)}$, $m = i, j$, are derived from (4.31) and

$$\begin{aligned}
\left(\sum_{\ell=1}^{3} u_{\ell,e_{ij}} \left\langle \frac{\partial \left(\tau_\ell\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \left(\rho u\right)_j}, n \right\rangle\right) &= \frac{\mu_{\mathrm{eff},e_{ij}}}{\mathrm{dist}(e_{ij})} \frac{1}{\rho_j} \begin{pmatrix} N_1 u_{1,e_{ij}} + \frac{1}{3} N_{12} u_{2,e_{ij}} + \frac{1}{3} N_{13} u_{3,e_{ij}} \\ \frac{1}{3} N_{12} u_{1,e_{ij}} + N_2 u_{2,e_{ij}} + \frac{1}{3} N_{23} u_{3,e_{ij}} \\ \frac{1}{3} N_{13} u_{1,e_{ij}} + \frac{1}{3} N_{23} u_{2,e_{ij}} + N_3 u_{3,e_{ij}} \end{pmatrix}^T, \\
\left(\sum_{\ell=1}^{3} u_{\ell,e_{ij}} \left\langle \frac{\partial \left(\tau_\ell\right)_{e_{ij}}^{\mathrm{TSL}}}{\partial \left(\rho u\right)_i}, n \right\rangle\right) &= -\frac{\mu_{\mathrm{eff},e_{ij}}}{\mathrm{dist}(e_{ij})} \frac{1}{\rho_i} \begin{pmatrix} N_1 u_{1,e_{ij}} + \frac{1}{3} N_{12} u_{2,e_{ij}} + \frac{1}{3} N_{13} u_{3,e_{ij}} \\ \frac{1}{3} N_{12} u_{1,e_{ij}} + N_2 u_{2,e_{ij}} + \frac{1}{3} N_{23} u_{3,e_{ij}} \\ \frac{1}{3} N_{13} u_{1,e_{ij}} + \frac{1}{3} N_{23} u_{2,e_{ij}} + N_3 u_{3,e_{ij}} \end{pmatrix}^T, \\
\frac{\partial \left\langle \left(q\right)_{e_{ij}}^{\mathrm{TSL}}, n \right\rangle}{\partial \left(\rho u\right)_j} &= \frac{\kappa_{\mathrm{eff},e_{ij}}}{\mathrm{dist}(e_{ij})} \frac{(1-\gamma)}{\rho_j} \begin{pmatrix} u_{j,1} \\ u_{j,2} \\ u_{j,3} \end{pmatrix}^T, \\
\frac{\partial \left\langle \left(q\right)_{e_{ij}}^{\mathrm{TSL}}, n \right\rangle}{\partial \left(\rho u\right)_i} &= \frac{\kappa_{\mathrm{eff},e_{ij}}}{\mathrm{dist}(e_{ij})} \frac{(\gamma-1)}{\rho_i} \begin{pmatrix} u_{i,1} \\ u_{i,2} \\ u_{i,3} \end{pmatrix}^T.
\end{aligned}$$

□

The approximate derivatives (4.34) and (4.35) represent a suitable expression which can be used for the design of a preconditioner. The complexity to generate these terms is acceptable and due to their explicit delineation these operators can be implemented straightforward.

4.3.2 Eigendecomposition of viscous flux Jacobian

For a stabilization of the solution method suggested in Chapter 5 an eigendecomposition, in particular the eigenvalues of the simplified viscous flux Jacobians (4.34) and (4.35) are of interest. These are given in the following theorem.

Theorem 4.3.4 *The derivative matrices of the viscous flux Jacobian for a* TSL *and constant* $\mu_{\mathrm{eff},e_{ij}}$ *given by (4.34)and (4.35) have the following eigenvalues:*

$$\begin{aligned}
\left(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\right)_i &= \frac{\mu_{\mathrm{eff},e_{ij}}}{\mathrm{dist}(e_{ij})} \frac{1}{\rho_i} \left(0, \frac{4}{3}, 1, 1, \frac{\kappa_{\mathrm{eff},e_{ij}}(\gamma-1)}{\mu_{\mathrm{eff},e_{ij}}}\right), \\
\left(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\right)_j &= -\frac{\mu_{\mathrm{eff},e_{ij}}}{\mathrm{dist}(e_{ij})} \frac{1}{\rho_j} \left(0, \frac{4}{3}, 1, 1, \frac{\kappa_{\mathrm{eff},e_{ij}}(\gamma-1)}{\mu_{\mathrm{eff},e_{ij}}}\right).
\end{aligned}$$

Proof: From representation (4.34) the eigenvalues

$$\left(\lambda_1\right)_j = 0 \qquad \text{and} \qquad \left(\lambda_5\right)_j = \frac{\kappa_{\mathrm{eff}}}{\mathrm{dist}(e_{ij})} \frac{\gamma-1}{\rho_j}$$

are obvious. To determine the remaining eigenvalues system (4.34) can be reduced to

$$\begin{pmatrix} N_1 & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} \\ \frac{1}{3}N_{12} & N_2 & \frac{1}{3}N_{23} \\ \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & N_3 \end{pmatrix} = \begin{pmatrix} 1+\frac{1}{3}n_1^2 & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} \\ \frac{1}{3}N_{12} & 1+\frac{1}{3}n_2^2 & \frac{1}{3}N_{23} \\ \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & 1+\frac{1}{3}n_3^2 \end{pmatrix}.$$

Therefore we need to determine non-trivial solutions of the linear system

$$\left[\lambda_j \mathbf{I} - \begin{pmatrix} 1+\frac{1}{3}n_1^2 & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} \\ \frac{1}{3}N_{12} & 1+\frac{1}{3}n_2^2 & \frac{1}{3}N_{23} \\ \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & 1+\frac{1}{3}n_3^2 \end{pmatrix}\right] x = 0.$$

Choosing $\lambda_j = 1$ yields the linear system and its equivalent formulation

$$\begin{pmatrix} \frac{1}{3}n_1^2 & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} \\ \frac{1}{3}N_{12} & \frac{1}{3}n_2^2 & \frac{1}{3}N_{23} \\ \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & \frac{1}{3}n_3^2 \end{pmatrix} x = 0 \quad \Leftrightarrow \quad \begin{pmatrix} n_1^2 & N_{12} & N_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x = 0.$$

Two non trivial linear independent solutions of this system are

$$x = (-(n_2+n_3), n_1, n_1) \quad \text{and} \quad x = (n_3 - n_2, n_1, -n_1). \tag{4.38}$$

Choosing $\lambda_j = 4/3$ yields the linear system and its equivalent formulation

$$\begin{pmatrix} \frac{1}{3}n_1^2 - \frac{1}{3} & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} \\ \frac{1}{3}N_{12} & \frac{1}{3}n_2^2 - \frac{1}{3} & \frac{1}{3}N_{23} \\ \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & \frac{1}{3}n_3^2 - \frac{1}{3} \end{pmatrix} x = 0 \quad \Leftrightarrow \quad \begin{pmatrix} n_1^2 - 1 & N_{12} & N_{13} \\ 0 & n_3^2 & n_2 n_3 \\ 0 & 0 & 0 \end{pmatrix} x = 0.$$

A non trivial solution of this system is given by

$$x = (n_1, n_2, n_3)\,. \tag{4.39}$$

The eigenvalues for (4.35) follow line by line from the computations above. This proves the assertion.

□

4.4 Derivative of turbulence models

To create an overall implicit solution method we additionally need the derivatives for the turbulence models. It is our approach to solve the original system of algebraic equations (3.95)) in a loosely coupled way, which finally yields the system of still time dependent equations (3.96). Being only interested in steady state solutions, we finally ended up with (3.97).

As a consequence, only the derivatives of the algebraic equations representing the turbulence models $\mathbf{R}_{\text{turb}}$ with respect to the turbulence flow variables $\mathbf{W}_t$ are required. And vice versa, the derivatives of the algebraic equations representing the

mean flow equations $\mathbf{R}_{\text{mean}}$ are only required with respect to $\mathbf{W}$. The latter were already presented in the Sections 4.2 and 4.3.

Hence, the organization of this section is as follows. First, we present the derivatives of the Spalart-Allmaras model with respect to $\tilde{\nu}$, and second the the derivatives of the $k\omega$-model with respect to k and ω. Again as in Section 4.3, to shorten the notation we neglect the edge (face) identifier e_{ij} for the normal, that is we write $n_{e_{ij}} = n = (n_1, n_2, n_3)^T$.

4.4.1 Derivative of Spalart-Allmaras model

We compute the derivatives for the Spalart-Allmaras model in the chronology, first the convective, second the viscous part and finally the source terms. The numerical flux function (3.72) is expressed by

$$\mathcal{H}_{\text{SA}}\left((\tilde{\nu}_i, W_i), (\tilde{\nu}_j, W_j), n\right) = \frac{1}{2}\left[\tilde{\nu}_i \langle u_i, n\rangle + \tilde{\nu}_j \langle u_j, n\rangle\right] - \frac{1}{2}|V_{ij,\text{Roe}}|\left(\tilde{\nu}_j - \tilde{\nu}_i\right).$$

Therefore, straightforward computations give

$$\frac{\partial \mathcal{H}_{\text{SA}}}{\partial \tilde{\nu}_m} = \frac{1}{2}\begin{cases} \langle u_i, n\rangle + |V_{ij,\text{Roe}}|, & m = i, \\ \langle u_j, n\rangle - |V_{ij,\text{Roe}}|, & m = j, \\ 0, & m \neq i, \quad m \neq j. \end{cases}$$

To compute the derivative for the viscous contribution we write down the discretized form explicitly,

$$\begin{aligned} \left\langle f^{\text{TSL}}_{v,\text{SA}}, n\right\rangle &= \frac{1}{\sigma}\begin{cases} \left(\nu_{l,e_{ij}} + \tilde{\nu}_{e_{ij}}\right)\sum_{k=1}^{3}\frac{n_k^2}{\text{dist}(e_{ij})}\left(\tilde{\nu}_j - \tilde{\nu}_i\right), & \tilde{\nu}_{e_{ij}} \geq 0, \\ \left(\nu_{l,e_{ij}} + f_{n,e_{ij}}\tilde{\nu}_{e_{ij}}\right)\sum_{k=1}^{3}\frac{n_k^2}{\text{dist}(e_{ij})}\left(\tilde{\nu}_j - \tilde{\nu}_i\right), & \tilde{\nu}_{e_{ij}} < 0, \end{cases} \\ &= \frac{1}{\sigma}\frac{\left(\tilde{\nu}_j - \tilde{\nu}_i\right)}{\text{dist}(e_{ij})}\begin{cases} \left(\nu_{l,e_{ij}} + \tilde{\nu}_{e_{ij}}\right), & \tilde{\nu}_{e_{ij}} \geq 0, \\ \left(\nu_{l,e_{ij}} + f_{n,e_{ij}}\tilde{\nu}_{e_{ij}}\right), & \tilde{\nu}_{e_{ij}} < 0. \end{cases} \end{aligned}$$

Since the laminar kinematic viscosity ν_l introduced in (2.13) (see also (3.76)) does not depend on $\tilde{\nu}$ we obtain

$$\frac{\partial \nu_{l,e_{ij}}}{\partial \tilde{\nu}_m} = 0 \qquad \text{for all} \qquad m = 1, \ldots, N_{\text{elem}}. \tag{4.40}$$

Using the definition (3.77) for $\tilde{\nu}_{e_{ij}}$ we obtain

$$\frac{\partial \tilde{\nu}_{e_{ij}}}{\partial \tilde{\nu}_m} = \begin{cases} \frac{1}{2}, & m = i, j, \\ 0 & \text{else.} \end{cases} \tag{4.41}$$

To compute the derivative of $f_{n,e_{ij}}$ we define

$$\begin{aligned} f^{\text{nom}}_{n,e_{ij}} &:= c_{n1} + \frac{1}{2}\left(\chi\left(\tilde{\nu}_i, W_i\right)^3 + \chi\left(\tilde{\nu}_j, W_j\right)^3\right), \\ f^{\text{denom}}_{n,e_{ij}} &:= c_{n1} - \frac{1}{2}\left(\chi\left(\tilde{\nu}_i, W_i\right)^3 + \chi\left(\tilde{\nu}_j, W_j\right)^3\right), \end{aligned}$$

to conclude

$$\begin{aligned}\frac{\partial f^{\text{nom}}_{n,e_{ij}}}{\partial \tilde{\nu}_m} &= \frac{1}{2}\frac{\partial}{\partial \tilde{\nu}_m}\left(\chi_i^3+\chi_j^3\right) = \frac{1}{2}\left(\frac{\partial \chi\left(\tilde{\nu}_i,W_i\right)^3}{\partial \tilde{\nu}_m}+\frac{\partial \chi\left(\tilde{\nu}_j,W_j\right)^3}{\partial \tilde{\nu}_m}\right)\\ &= \frac{1}{2}\left(\chi\left(\tilde{\nu}_i,W_i\right)^2\frac{\partial \chi\left(\tilde{\nu}_i,W_i\right)}{\partial \tilde{\nu}_m}+\chi\left(\tilde{\nu}_j,W_j\right)^2\frac{\partial \chi\left(\tilde{\nu}_j,W_j\right)}{\partial \tilde{\nu}_m}\right),\\ \frac{\partial f^{\text{denom}}_{n,e_{ij}}}{\partial \tilde{\nu}_m} &= -\frac{\partial f^{\text{nom}}_{n,e_{ij}}}{\partial \tilde{\nu}_m}\end{aligned}$$

where

$$\frac{\partial \chi\left(\tilde{\nu}_\ell,W_\ell\right)}{\partial \tilde{\nu}_m} = \begin{cases}\left(\nu_l\left(W_i\right)\right)^{-1}, & m=\ell=i,\\ \left(\nu_l\left(W_j\right)\right)^{-1}, & m=\ell=j,\\ 0 & m\neq\ell,\end{cases}$$

such that the derivative of $f_{n,e_{ij}}$ may be given by

$$\frac{\partial f_{n,e_{ij}}}{\partial \tilde{\nu}_m} = \frac{\frac{\partial f^{\text{nom}}_{n,e_{ij}}}{\partial \tilde{\nu}_m} f^{\text{denom}}_{n,e_{ij}} - f^{\text{nom}}_{n,e_{ij}}\frac{\partial f^{\text{denom}}_{n,e_{ij}}}{\partial \tilde{\nu}_m}}{\left(f^{\text{denom}}_{n,e_{ij}}\right)^2} = \frac{\frac{\partial f^{\text{nom}}_{n,e_{ij}}}{\partial \tilde{\nu}_m}\left(f^{\text{denom}}_{n,e_{ij}}+f^{\text{nom}}_{n,e_{ij}}\right)}{\left(f^{\text{denom}}_{n,e_{ij}}\right)^2} = c_{n1}\frac{\frac{\partial f^{\text{nom}}_{n,e_{ij}}}{\partial \tilde{\nu}_m}}{\left(f^{\text{denom}}_{n,e_{ij}}\right)^2}.$$

Then we get for the derivative of the viscous part

$$\begin{aligned}&\frac{\partial\left\langle f^{\text{TSL}}_{v,\text{SA}},n\right\rangle}{\partial \tilde{\nu}_i}\\ &= \frac{1}{\sigma \text{dist}(e_{ij})}\begin{cases}-\left(\nu_{l,e_{ij}}+\tilde{\nu}_{e_{ij}}\right)+\left(\tilde{\nu}_j-\tilde{\nu}_i\right)\frac{\partial \tilde{\nu}_{e_{ij}}}{\partial \tilde{\nu}_i}, & \tilde{\nu}_{e_{ij}}\geq 0,\\ -\left(\nu_{l,e_{ij}}+f_{n,e_{ij}}\tilde{\nu}_{e_{ij}}\right)+\left(\tilde{\nu}_j-\tilde{\nu}_i\right)\left[f_{n,e_{ij}}\frac{\partial \tilde{\nu}_{e_{ij}}}{\partial \tilde{\nu}_i}+\tilde{\nu}_{e_{ij}}\frac{\partial f_{n,e_{ij}}}{\partial \tilde{\nu}_i}\right], & \tilde{\nu}_{e_{ij}}<0,\end{cases}\end{aligned}$$

and

$$\begin{aligned}&\frac{\partial\left\langle f^{\text{TSL}}_{v,\text{SA}},n\right\rangle}{\partial \tilde{\nu}_j}\\ &= \frac{1}{\sigma \text{dist}(e_{ij})}\begin{cases}\left(\nu_{l,e_{ij}}+\tilde{\nu}_{e_{ij}}\right)+\left(\tilde{\nu}_j-\tilde{\nu}_i\right)\frac{\partial \tilde{\nu}_{e_{ij}}}{\partial \tilde{\nu}_j}, & \tilde{\nu}_{e_{ij}}\geq 0,\\ \left(\nu_{l,e_{ij}}+f_{n,e_{ij}}\tilde{\nu}_{e_{ij}}\right)+\left(\tilde{\nu}_j-\tilde{\nu}_i\right)\left[f_{n,e_{ij}}\frac{\partial \tilde{\nu}_{e_{ij}}}{\partial \tilde{\nu}_j}+\tilde{\nu}_{e_{ij}}\frac{\partial f_{n,e_{ij}}}{\partial \tilde{\nu}_j}\right], & \tilde{\nu}_{e_{ij}}<0.\end{cases}\end{aligned}$$

Finally, the derivatives of the discretized source terms (3.79a), (3.79b) and (3.79c) are required. Note that the production and destruction terms only depend on the variable ν_i. That is all other derivatives vanish,

$$\begin{aligned}\frac{\partial Pr_{\text{SA}}\left(\tilde{\nu}_i,W_i,W_{j,j\in\mathcal{N}(i)}\right)}{\partial \tilde{\nu}_m} &= 0, \qquad m\neq i,\\ \frac{\partial De_{\text{SA}}\left(\tilde{\nu}_i,W_i,W_{j,j\in\mathcal{N}(i)}\right)}{\partial \tilde{\nu}_m} &= 0, \qquad m\neq i.\end{aligned}$$

To shorten the notation, in the following we will list all required derivatives, without denoting that all derivatives with respect $m \neq i$ vanish,

$$\begin{aligned}
\frac{\partial f_{\mathrm{t}_2}}{\partial \tilde{\nu}_i} &= -2c_{\mathrm{t}_3}c_{\mathrm{t}_4}\exp\left(-c_{\mathrm{t}_4}\chi^2\right)\chi\frac{\partial \chi}{\partial \tilde{\nu}_i} = -2c_{\mathrm{t}_4}f_{\mathrm{t}_2}\chi\frac{\partial \chi}{\partial \tilde{\nu}_i},\\
\frac{\partial f_{\mathrm{v}_1}}{\partial \tilde{\nu}_i} &= \frac{1}{(\chi^3+c_{\mathrm{v}_1}^3)^2}\left[\left(\chi^3+c_{\mathrm{v}_1}^3\right)3\chi^2\frac{\partial \chi}{\partial \tilde{\nu}_i} - 3\chi^5\frac{\partial \chi}{\partial \tilde{\nu}_i}\right] = \frac{3c_{\mathrm{v}_1}^3\chi^2}{(\chi^3+c_{\mathrm{v}_1}^3)^2}\frac{\partial \chi}{\partial \tilde{\nu}_i},\\
\frac{\partial f_{\mathrm{v}_2}}{\partial \tilde{\nu}_i} &= -\frac{1}{(1+\chi f_{\mathrm{v}_1})^2}\left[(1+\chi f_{\mathrm{v}_1})\frac{\partial \chi}{\partial \tilde{\nu}_i} - \chi\left(f_{\mathrm{v}_1}\frac{\partial \chi}{\partial \tilde{\nu}_i} + \chi\frac{\partial f_{\mathrm{v}_1}}{\partial \tilde{\nu}_i}\right)\right]\\
&= \frac{1}{(1+\chi f_{\mathrm{v}_1})^2}\left(\chi^2\frac{\partial f_{\mathrm{v}_1}}{\partial \tilde{\nu}_i} - \frac{\partial \chi}{\partial \tilde{\nu}_i}\right).
\end{aligned}$$

To compute the derivative of $\tilde{S}$ note that

$$\frac{\partial S}{\partial \tilde{\nu}_i} = 0, \qquad i = 1, \ldots, N_{\mathrm{elem}}.$$

Hence, in the case $\overline{S} \geq -c_{v2}S$ we have

$$\frac{\partial \tilde{S}}{\partial \tilde{\nu}_i} = \frac{\partial \overline{S}}{\partial \tilde{\nu}_i} = \frac{1}{\kappa^2 d^2}\left(f_{\mathrm{v}_2} + \tilde{\nu}\frac{\partial f_{\mathrm{v}_2}}{\partial \tilde{\nu}_i}\right),$$

and for the other case $\overline{S} < -c_{v2}S$ we define

$$\begin{aligned}
\overline{S}^{\mathrm{nom}} &:= S\left(c_{v2}^2 S + c_{v3}\overline{S}\right)\\
\overline{S}^{\mathrm{denom}} &:= \left(c_{v3} - 2c_{v2}\right)S - \overline{S}.
\end{aligned}$$

to compute

$$\frac{\partial \tilde{S}}{\partial \tilde{\nu}_i} = \frac{\overline{S}^{\mathrm{denom}}\frac{\partial \overline{S}^{\mathrm{nom}}}{\partial \tilde{\nu}_i} - \overline{S}^{\mathrm{nom}}\frac{\partial \overline{S}^{\mathrm{denom}}}{\partial \tilde{\nu}_i}}{\left(\overline{S}^{\mathrm{denom}}\right)^2}$$

where

$$\frac{\partial \overline{S}^{\mathrm{nom}}}{\partial \tilde{\nu}_i} = c_{v3}S\frac{\partial \overline{S}}{\partial \tilde{\nu}_i} \qquad \text{and} \qquad \frac{\partial \overline{S}^{\mathrm{denom}}}{\partial \tilde{\nu}_i} = -\frac{\partial \overline{S}}{\partial \tilde{\nu}_i}.$$

The derivatives of r, g and f_w are given by

$$\begin{aligned}
\frac{\partial r}{\partial \tilde{\nu}_i} &= \begin{cases} \frac{1}{\kappa^2 d^2 \tilde{S}^2}\left(\tilde{S} - \tilde{\nu}\frac{\partial \tilde{S}}{\partial \tilde{\nu}_i}\right), & \frac{\tilde{\nu}_i}{\kappa^2 d^2 \tilde{S}} \leq 10 \\ 0, & \text{else,} \end{cases}\\
\frac{\partial g}{\partial \tilde{\nu}_i} &= \frac{\partial r}{\partial \tilde{\nu}_i} + c_{\mathrm{w}_2}\left[6r^5\frac{\partial r}{\partial \tilde{\nu}_i} - \frac{\partial r}{\partial \tilde{\nu}_i}\right],\\
\frac{\partial f_w}{\partial \tilde{\nu}_i} &= \left[\frac{1+c_{\mathrm{w}_3}^6}{g^6+c_{\mathrm{w}_3}^6}\right]^{1/6}\frac{\partial g}{\partial \tilde{\nu}_i} + \frac{g}{6}\left[\frac{1+c_{\mathrm{w}_3}^6}{g^6+c_{\mathrm{w}_3}^6}\right]^{-5/6}\left(-\frac{6(1+c_{\mathrm{w}_3}^6)g^5\frac{\partial g}{\partial \tilde{\nu}_i}}{(g^6+c_{\mathrm{w}_3}^6)^2}\right).
\end{aligned}$$

Finally, the derivatives of production and destruction can be computed by

$$\frac{\partial Pr_{\mathrm{SA}}}{\partial \tilde{\nu}_i} = \begin{cases} c_{\mathrm{b}_1}\left[-\tilde{S}\tilde{\nu}_i\frac{\partial f_{\mathrm{t}_2}}{\partial \tilde{\nu}_i} + (1-f_{\mathrm{t}_2})\tilde{\nu}_i\frac{\partial \tilde{S}}{\partial \tilde{\nu}_i} + (1-f_{\mathrm{t}_2})\tilde{S}\right], & \tilde{\nu}_i \geq 0, \\ c_{\mathrm{b}_1}\left(1-c_{t3}\right)S\frac{\partial \tilde{\nu}}{\partial \tilde{\nu}_i}, & \tilde{\nu}_i < 0, \end{cases}$$

$$\frac{\partial De_{\mathrm{SA}}}{\partial \tilde{\nu}_i} = \begin{cases} \left(\frac{\tilde{\nu}}{d}\right)^2\left(c_{\mathrm{w}_1}\frac{\partial f_{\mathrm{w}}}{\partial \tilde{\nu}_i} - \frac{c_{b_1}}{\kappa^2}\frac{\partial f_{\mathrm{t}_2}}{\partial \tilde{\nu}_i}\right) + \left(c_{\mathrm{w}_1}f_{\mathrm{w}} - \frac{c_{\mathrm{b}_1}}{\kappa^2}f_{\mathrm{t}_2}\right)\frac{2\tilde{\nu}_i}{d^2}, & \tilde{\nu}_i \geq 0 \\ -2c_{w1}\frac{\tilde{\nu}_i}{d^2}, & \tilde{\nu}_i < 0. \end{cases}$$

The diffusion source term of the Spalart-Allmaras model reads in discretized form

$$Di_{\mathrm{SA}} = \frac{c_{b2}}{\sigma}\sum_{k=1}^{3}\left[\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)_{D_i}^{\mathrm{GG}}\right]^2.$$

An application of the chain rule yields

$$\frac{\partial Di_{\mathrm{SA}}}{\partial \tilde{\nu}_m} = \frac{2c_{b2}}{\sigma}\sum_{k=1}^{3}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)_{D_i}^{\mathrm{GG}}\frac{\partial}{\partial \tilde{\nu}_m}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)_{D_i}^{\mathrm{GG}}.$$

Using formula (3.60) of the approximate gradient

$$\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} = \frac{1}{\mathrm{vol}\left(D_i\right)}\sum_{j\in\mathcal{N}(i)}\mathrm{svol}(e_{ij})\frac{n_{k,e_{ij}}}{2}\left(\tilde{\nu}_i+\tilde{\nu}_j\right),$$

we get for its derivative

$$\frac{\partial}{\partial \tilde{\nu}_m}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\right)_{D_i}^{\mathrm{GG}} = \frac{1}{\mathrm{vol}\left(D_i\right)}\begin{cases} \sum_{j\in\mathcal{N}(i)}\mathrm{svol}(e_{ij})\frac{n_{k,e_{ij}}}{2}, & m=i, \\ \mathrm{svol}(e_{im})\frac{n_{k,e_{im}}}{2}, & m\in\mathcal{N}(i), \\ 0, & \text{else.} \end{cases}$$

It is worthwhile to note that the derivative of Di_{SA} also gives contributions to off-diagonal terms, which is not true for the production and destruction. This gives all required terms for the derivative of the Spalart-Allmaras turbulence model.

4.4.2 Derivative of $k\omega$-model

The computation of the derivative for the $k\omega$-model is far simpler compared to the Spalart-Allmaras model. The flux function (3.73) given in Definition 3.5.2 is straightforward differentiated,

$$\frac{\partial \mathcal{H}_{(k,\omega)}}{\partial(k_i,\omega_i)} = \begin{pmatrix} \begin{cases} \left\langle u_{e_{ij}}, n\right\rangle, & \left\langle u_{e_{ij}}, n\right\rangle \geq 0, \\ 0, & \left\langle u_{e_{ij}}, n\right\rangle < 0 \end{cases} & 0 \\ 0 & \begin{cases} \left\langle u_{e_{ij}}, n\right\rangle, & \left\langle u_{e_{ij}}, n\right\rangle \geq 0, \\ 0, & \left\langle u_{e_{ij}}, n\right\rangle < 0 \end{cases} \end{pmatrix},$$

$$\frac{\partial \mathcal{H}_{(k,\omega)}}{\partial(k_j,\omega_j)} = \begin{pmatrix} \begin{cases} 0, & \left\langle u_{e_{ij}}, n\right\rangle < 0 \\ \left\langle u_{e_{ij}}, n\right\rangle, & \left\langle u_{e_{ij}}, n\right\rangle \geq 0, \end{cases} & 0 \\ 0 & \begin{cases} 0, & \left\langle u_{e_{ij}}, n\right\rangle < 0 \\ \left\langle u_{e_{ij}}, n\right\rangle, & \left\langle u_{e_{ij}}, n\right\rangle \geq 0, \end{cases} \end{pmatrix}.$$

Following the presentation for the Spalart-Allmaras model, we write down the discretized form of the viscous contribution explicitly,

$$\begin{aligned}\left\langle \tilde{f}^{\mathrm{TSL}}_{v,(k,\omega)}, n \right\rangle &= \begin{pmatrix} \left(\Gamma(T)_{e_{ij}} + \sigma_k \left(\frac{k}{\omega}\right)_{e_{ij}}\right) \sum_{\ell=1}^{3} \frac{n_\ell^2}{\mathrm{dist}(e_{ij})} (k_j - k_i) \\ \left(\Gamma(T)_{e_{ij}} + \sigma_\omega \left(\frac{k}{\omega}\right)_{e_{ij}}\right) \sum_{\ell=1}^{3} \frac{n_\ell^2}{\mathrm{dist}(e_{ij})} (\omega_j - \omega_i) \end{pmatrix} \\ &= \frac{1}{\mathrm{dist}(e_{ij})} \begin{pmatrix} \left(\Gamma(T)_{e_{ij}} + \sigma_k \left(\frac{k}{\omega}\right)_{e_{ij}}\right) (k_j - k_i) \\ \left(\Gamma(T)_{e_{ij}} + \sigma_\omega \left(\frac{k}{\omega}\right)_{e_{ij}}\right) (\omega_j - \omega_i) \end{pmatrix}.\end{aligned}$$

Since $\Gamma(T)_{e_{ij}}$ defined in (3.78) does not depend on (k_i, ω_i) for all $i = 1, \ldots, N_{\mathrm{elem}}$ due to our definition of temperature (2.4) and the equation of state (2.3), we have

$$\frac{\partial \Gamma}{\partial (k_i, \omega_i)} = 0.$$

Hence, we obtain

$$\frac{\partial \left\langle \tilde{f}^{\mathrm{TSL}}_{v,(k,\omega)}, n \right\rangle}{\partial (k_i, \omega_i)} = \frac{1}{\mathrm{dist}(e_{ij})} \begin{pmatrix} \sigma_k \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial k_i}\right) (k_j - k_i) - \left(\Gamma(T)_{e_{ij}} + \sigma_k \left(\frac{k}{\omega}\right)_{e_{ij}}\right) & \sigma_k \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial \omega_i}\right) (k_j - k_i) \\ \sigma_\omega \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial k_i}\right) (\omega_j - \omega_i) & \sigma_\omega \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial \omega_i}\right) (\omega_j - \omega_i) - \left(\Gamma(T)_{e_{ij}} + \sigma_\omega \left(\frac{k}{\omega}\right)_{e_{ij}}\right) \end{pmatrix}$$

and

$$\frac{\partial \left\langle \tilde{f}^{\mathrm{TSL}}_{v,(k,\omega)}, n \right\rangle}{\partial (k_j, \omega_j)} = \frac{1}{\mathrm{dist}(e_{ij})} \begin{pmatrix} \sigma_k \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial k_j}\right) (k_j - k_i) + \left(\Gamma(T)_{e_{ij}} + \sigma_k \left(\frac{k}{\omega}\right)_{e_{ij}}\right) & \sigma_k \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial \omega_j}\right) (k_j - k_i) \\ \sigma_\omega \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial k_j}\right) (\omega_j - \omega_i) & \sigma_\omega \left(\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial \omega_j}\right) (\omega_j - \omega_i) + \left(\Gamma(T)_{e_{ij}} + \sigma_\omega \left(\frac{k}{\omega}\right)_{e_{ij}}\right) \end{pmatrix},$$

where

$$\frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial (k_i, \omega_i)} = \frac{1}{2} \begin{pmatrix} \frac{1}{\omega_i}, -\frac{k_i}{\omega_i^2} \end{pmatrix} \qquad \text{and} \qquad \frac{\partial \left(\frac{k}{\omega}\right)_{e_{ij}}}{\partial (k_j, \omega_j)} = \frac{1}{2} \begin{pmatrix} \frac{1}{\omega_j}, -\frac{k_j}{\omega_j^2} \end{pmatrix}.$$

The derivative of the discretized source terms (3.80) is also straightforward. Note that with respect to the variables (k_i, ω_i), we can directly conclude that

$$\begin{aligned}
\frac{\partial Pr_{k,(k,\omega)}}{\partial(k_m,\omega_m)} &= \begin{cases} \left(\frac{2}{\omega_i}\overline{\mathcal{S}}\left(u\right)_{D_i} \otimes \left(\frac{\partial u}{\partial x}\right)_{D_i}, -\frac{2}{\omega_i^2}\overline{\mathcal{S}}\left(u\right)_{D_i} \otimes \left(\frac{\partial u}{\partial x}\right)_{D_i}\right) & m = i, \\ (0,0), & m \neq i, \end{cases} \\
\frac{\partial De_{k,(k,\omega)}\left(k_i,\omega_i\right)}{\partial(k_m,\omega_m)} &= \begin{cases} \left(\beta^*\omega_i, \beta^* k_i\right), & m = i, \\ (0,0), & m \neq i, \end{cases} \\
\frac{\partial Pr_{\omega,(k,\omega)}}{\partial(k_m,\omega_m)} &= (0,0), \qquad m = 1,\ldots,N_{\text{elem}} \\
\frac{\partial De_{\omega,(k,\omega)}}{\partial\left(k_m,\omega_m\right)} &= \begin{cases} \left(0, 2\beta\omega_i\right), & m = i, \\ (0,0), & m \neq i. \end{cases}
\end{aligned}$$

Finally, note that for the computation of the derivative of the production term the limitation (3.81) needs to be incorporated, as well as the scaling factor ω_{sc} needs to be included. The latter remark is also true for the viscous part of the derivative.

4.4.3 Structure of derivative for turbulence models

Let us close this section with an important remark. Both, the derivatives for the Spalart-Allmaras model as well as for the $k\omega$-model are based on a compact stencil. Hence, the nonzero entries in the (block) CSR matrix are determined by i and $\mathcal{N}(i)$, and the derivative can be computed by loop over all edges (faces).

Corollary 4.4.1 *Consider a 3D hexahedral structured mesh representing N degrees of freedom. Then the memory requirements for $\frac{d\mathbf{R}_{\text{SA}}^{\text{comp}}}{d\mathbf{W}}$ and for $\frac{d\mathbf{R}_{(k,\omega)}^{\text{comp}}}{d\mathbf{W}}$ are*

$$56 \cdot N\,Bytes \qquad and \qquad 224 \cdot N\,Bytes,$$

The proof follows line by line the proofs Lemma 4.1.6 and Lemma 4.2 taking into account that the block size for $\frac{d\mathbf{R}_{\text{SA}}^{\text{comp}}}{d\mathbf{W}}$ is 1×1 , and the block size for $\frac{d\mathbf{R}_{(k,\omega)}^{\text{comp}}}{d\mathbf{W}}$ is 2×2. □

Since, in particular, it is often a severe issue to approximate a solution of the turbulence flow equations in a robust and efficient way, it is one goal to care about good agreement of the exact derivative and a preconditioner used to improve the efficiency and robustness. To this end, within this these we have only considered compact discretization schemes for the convective part of the equations. In particular, considering only a TSL approximation of the gradients in the turbulence flow equations, we have exact derivatives for the turbulence flow equations considering only compact stencils. This is a concession between robustness of the solution method and accuracy of the discretization on the other hand.

4.4.4 Derivative of eddy viscosity

To complete the derivative of the viscous terms for the mean flow equations, it was mentioned in Section 4.3 that the derivative of the eddy viscosity μ_t is required. The derivative of κ_t is a consequence of μ_t due to (2.14). Note that both the Spalart-Allmaras model and the $k\omega$-model do not depend on the eddy viscosity. Both models are formulated without dependency of this deduced functions. Hence, we only state the derivatives of these functions with respect to the mean flow variables.

Due to (3.66) we have

$$\mu_{t,e_{ij}} = \frac{1}{2}\left(\mu_{t,i} + \mu_{t,j}\right) = \frac{1}{2}\left\{\mu_t\left(\left(W_t\right)_i, W_i\right) + \mu_t\left(\left(W_t\right)_j, W_j\right)\right\},$$

and hence

$$\frac{\partial \mu_{t,e_{ij}}}{\partial W_m} = \frac{1}{2}\begin{cases} \frac{\partial \mu_t\left(\left(W_t\right)_i, W_i\right)}{\partial W_i}, & m = i, \\ \frac{\partial \mu_t\left(\left(W_t\right)_j, W_j\right)}{\partial W_j}, & m = j, \\ 0, & m \neq i, m \neq j. \end{cases}$$

Using (2.20) for the Spalart-Allmaras model we get

$$\frac{\partial \mu_t\left(\left(W_t\right)_i, W_i\right)}{\partial W_i} = \begin{cases} \tilde{\nu}_i\left(f_{v1}\frac{\partial \rho_i}{\partial W_i} + \rho_i \frac{\partial f_{v1}}{\partial W_i}\right), & \tilde{\nu}_i \geq 0, \\ 0, & \tilde{\nu}_i < 0. \end{cases}$$

For the $k\omega$-model we get using (2.21)

$$\frac{\partial \mu_t\left(\left(W_t\right)_i, W_i\right)}{\partial W_i} = \frac{\sqrt{\gamma} M_\infty L}{Re} \rho \frac{k_i}{\omega_i} \frac{\partial \rho_i}{\partial W_i}.$$

Including these derivatives into the viscous part of the mean flow equations completes this contribution.

4.5 Derivative of boundary conditions

Since all boundary conditions are formulated as fluxes over the boundary edges $e_{i,\mathrm{bdry}}$, the derivatives can be computed in principle in the same way it is done for the inner fluxes. Only the extrapolation of the boundary values need to be incorporated into the formulation. Mathematically spoken, according to the considerations of Section 3.6 we assume that the boundary values are a function of the variables attached to the boundary element,

$$\begin{aligned} W_{i,\mathrm{bdry}} &= W_{i,\mathrm{bdry}}\left(W_i, W_{j,j\in\mathcal{N}(i)}\right), \\ \tilde{\nu}_{i,\mathrm{bdry}} &= \tilde{\nu}_{i,\mathrm{bdry}}\left(\tilde{\nu}_i, \tilde{\nu}_{j,j\in\mathcal{N}(i)}\right), \\ \left(k_{i,\mathrm{bdry}}, \omega_{i,\mathrm{bdry}}\right) &= \left(k_{i,\mathrm{bdry}}\left(k_i, k_{j,j\in\mathcal{N}(i)}\right), \omega_{i,\mathrm{bdry}}\left(\omega_i, \omega_{j,j\in\mathcal{N}(i)}\right)\right). \end{aligned}$$

Then, the derivative of the boundary flux is differentiated generally, using the derivatives for the inner flux functions, and an additional application of the chain rule,

$$\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}\left(W_i, W_{i,\mathrm{bdry}}, n_{i,\mathrm{bdry}}\right)}{\partial W_k} = \left(\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial\left(W_i, W_{i,\mathrm{bdry}}\right)}\right) \begin{pmatrix} \frac{\partial W_i}{\partial W_k} \\ \frac{\partial W_{i,\mathrm{bdry}}}{\partial W_k} \end{pmatrix}. \tag{4.42}$$

The first term $\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial\left(W_i, W_{i,\mathrm{bdry}}\right)}$ is already discussed in Section 4.2.1 and represents nothing else than the inner flux derivative. Hence, as mentioned above the inner flux derivative is therefore used to compute the derivative of the boundary flux. All, what is required for the boundary derivative, is the implementation of the matrix-matrix product, which can also be avoided due to

$$\frac{\partial W_i}{\partial W_k} = \begin{cases} \mathbf{I}, & k = i, \\ 0, & k \neq i, \end{cases}$$

and therefore

$$\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}\left(W_i, W_{i,\mathrm{bdry}}, n_{i,\mathrm{bdry}}\right)}{\partial W_k} = \begin{cases} \frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_i} + \frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_{i,\mathrm{bdry}}} \frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i}, & k = i, \\ \frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_{i,\mathrm{bdry}}} \frac{\partial W_{i,\mathrm{bdry}}}{\partial W_k}, & k \neq i. \end{cases}$$

Naturally, the same consideration holds true for the viscous flux,

$$\frac{\partial \left\langle f_v\left(W_i, W_{i,\mathrm{bdry}}\right), n_{i,\mathrm{bdry}}\right\rangle}{\partial W_k} = \left(\frac{\partial \left\langle f_v\left(W_i, W_{i,\mathrm{bdry}}\right), n_{i,\mathrm{bdry}}\right\rangle}{\partial\left(W_i, W_{i,\mathrm{bdry}}\right)}\right) \begin{pmatrix} \frac{\partial W_i}{\partial W_k} \\ \frac{\partial W_{i,\mathrm{bdry}}}{\partial W_k} \end{pmatrix},$$

and for the boundary conditions of the turbulence flow equations,

$$\begin{aligned} \frac{\partial \mathcal{H}_{\mathrm{SA}}}{\partial \tilde{\nu}_m} &= \frac{\partial \mathcal{H}_{\mathrm{SA}}}{\partial\left(\tilde{\nu}_i, \tilde{\nu}_{i,\mathrm{bdry}}\right)} \begin{pmatrix} \frac{\partial \tilde{\nu}_i}{\partial \tilde{\nu}_m} \\ \frac{\partial \tilde{\nu}_{i,\mathrm{bdry}}}{\partial \tilde{\nu}_m} \end{pmatrix} \\ \frac{\partial \mathcal{H}_{(k,\omega)}}{\partial\left(k_m, \omega_m\right)} &= \frac{\partial \mathcal{H}_{(k,\omega)}}{\partial\left(\left(k_i, \omega_i\right), \left(k_{i,\mathrm{bdry}}\omega_{i,\mathrm{bdry}}\right)\right)} \begin{pmatrix} \frac{\partial (k_i,\omega_i)}{\partial (k_m,\omega_m)} \\ \frac{\partial \left(k_{i,\mathrm{bdry}},\omega_{i,\mathrm{bdry}}\right)}{\partial (k_m,\omega_m)} \end{pmatrix}, \end{aligned}$$

and

$$\begin{aligned} \frac{\partial \left\langle f_{v,\mathrm{SA}}, n_{i,\mathrm{bdry}}\right\rangle}{\partial \tilde{\nu}_m} &= \frac{\partial \left\langle f_{v,\mathrm{SA}}, n_{i,\mathrm{bdry}}\right\rangle}{\partial\left(\tilde{\nu}_i, \tilde{\nu}_{i,\mathrm{bdry}}\right)} \begin{pmatrix} \frac{\partial \tilde{\nu}_i}{\partial \tilde{\nu}_m} \\ \frac{\partial \tilde{\nu}_{i,\mathrm{bdry}}}{\partial \tilde{\nu}_m} \end{pmatrix}, \\ \frac{\partial \left\langle \tilde{f}_{v,(k,\omega)}, n_{i,\mathrm{bdry}}\right\rangle}{\partial\left(k_m, \omega_m\right)} &= \frac{\partial \left\langle \tilde{f}_{v,(k,\omega)}, n_{i,\mathrm{bdry}}\right\rangle}{\partial\left(\left(k_i, \omega_i\right), \left(k_{i,\mathrm{bdry}}\omega_{i,\mathrm{bdry}}\right)\right)} \begin{pmatrix} \frac{\partial (k_i,\omega_i)}{\partial (k_m,\omega_m)} \\ \frac{\partial \left(k_{i,\mathrm{bdry}},\omega_{i,\mathrm{bdry}}\right)}{\partial (k_m,\omega_m)} \end{pmatrix}. \end{aligned}$$

Farfield boundary condition. To apply these formulae for certain boundary conditions, consider for example, flux of the farfield boundary condition. Here the

outer states $W_\infty, \tilde{\nu}_\infty$ and $(k_\infty, \omega_\infty)$ do not depend on inner states. Hence, we obtain for the derivative

$$\frac{\partial W_\infty}{\partial W_m} = 0, \quad \frac{\partial \tilde{\nu}_\infty}{\partial \tilde{\nu}_m} = 0, \quad \frac{\partial (k_\infty, \omega_\infty)}{\partial (k_m, \omega_m)} = 0, \qquad m = 1, \ldots, N_{\text{elem}},$$

and therefore, for example,

$$\frac{\partial \mathcal{H}^{1st,\text{Roe}}(W_i, W_\infty, n_{i,\text{bdry}})}{\partial W_m} = \begin{cases} \frac{\partial \mathcal{H}^{1st,\text{Roe}}}{\partial W_i}, & m = i, \\ 0, & m \neq i. \end{cases}$$

The last conclusion can also be applied straightforward to the viscous flux as well as the fluxes for the turbulence flow equations.

Slip wall boundary condition. For the slip wall boundary condition we compute

$$\frac{\partial W_{i,\text{bdry}}}{\partial W_k} = \begin{cases} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1-2n_{1,\text{bdry}}^2 & 2n_{1,\text{bdry}}n_{2,\text{bdry}} & 2n_{1,\text{bdry}}n_{3,\text{bdry}} & 0 \\ 0 & 2n_{2,\text{bdry}}n_{1,\text{bdry}} & 1-2n_{2,\text{bdry}}^2 & 2n_{2,\text{bdry}}n_{3,\text{bdry}} & 0 \\ 0 & 2n_{3,\text{bdry}}n_{1,\text{bdry}} & 2n_{3,\text{bdry}}n_{2,\text{bdry}} & 1-2n_{3,\text{bdry}}^2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, & k = i, \\ 0, & k \neq i. \end{cases}$$

Since slip wall boundaries correspond in our context only to inviscid flows, we neglect turbulence here.

No-slip wall boundary condition. To compute the derivative for the no-slip wall boundary condition is more complicated. First of all note, that we suggested several variants to compute gradients for the viscous flux. Here we restrict ourselves to (3.84a)– (3.84c) in combination with (3.85a)– (3.85c). The principle to extend the derivative to other kind of gradients stays the same. To formulate the convective fluxes over the no-slip wall we used (3.82a)– (3.82c), and hence

$$\frac{\partial W_{i,\text{bdry}}}{\partial W_k} = \begin{cases} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, & k = i, \\ 0, & k \neq i. \end{cases}$$

In particular, an application of (4.8), (4.9) together with (4.42) yields

$$\begin{aligned}
& 2\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_i} \\
= \;& \frac{\partial \langle f_c, n\rangle}{\partial W_i} + \frac{\partial \langle f_c, n\rangle}{\partial W_{i,\mathrm{bdry}}}\frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i} \\
+ \;& \frac{\partial \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|}{\partial W_i}\left(W_{i,\mathrm{bdry}} - W_i\right) + \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|\frac{\partial}{\partial W_i}\left(W_{i,\mathrm{bdry}} - W_i\right) \\
+ \;& \left\{\frac{\partial \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|}{\partial W_{i,\mathrm{bdry}}}\left(W_{i,\mathrm{bdry}} - W_i\right) + \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|\frac{\partial}{\partial W_{i,\mathrm{bdry}}}\left(W_{i,\mathrm{bdry}} - W_i\right)\right\}\frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i} \\
= \;& \frac{\partial \langle f_c, n\rangle}{\partial W_i} + \frac{\partial \langle f_c, n\rangle}{\partial W_{i,\mathrm{bdry}}}\frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i} + \frac{\partial \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|}{\partial W_i}\left(W_{i,\mathrm{bdry}} - W_i\right) - \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right| \\
+ \;& \left\{\frac{\partial \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|}{\partial W_{i,\mathrm{bdry}}}\left(W_{i,\mathrm{bdry}} - W_i\right) + \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|\right\}\frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i}.
\end{aligned}$$

To construct a suitable solution method, according to Section 4.2.2, we may neglect terms including $\frac{\partial \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|}{\partial W_k}$. Then we obtain an approximate derivative by

$$\begin{aligned}
2\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_i} &\approx \frac{\partial \langle f_c, n\rangle}{\partial W_i} + \frac{\partial \langle f_c, n\rangle}{\partial W_{i,\mathrm{bdry}}}\frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i} - \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right| + \left|\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right|\frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i} \\
&= \frac{\partial \langle f_c, n\rangle}{\partial W_i} + \frac{\partial \langle f_c, n\rangle}{\partial W_{i,\mathrm{bdry}}}\frac{\partial W_{i,\mathrm{bdry}}}{\partial W_i} - 2\left(0, \left|\left(\mathbf{A}^{\mathrm{Roe}}_{i,\mathrm{bdry}}\right)_{k\ell}\right|_{1\le k\le 5, 2\le \ell\le 4}, 0\right).
\end{aligned}$$

For the derivative of the viscous flux we additionally need to incorporate (3.85a)–(3.85c) as well as (3.83). As a consequence, the viscous flux depends on W_i, $W_{i,\mathrm{bdry}}$ and $W_{i,n}$, where we have used the notation of (3.84a). Hence, differentiating (3.85a)–(3.85c) the equations (4.19)–(4.22) are replaced by

$$\begin{aligned}
\frac{\partial}{\partial W_m}\left(\frac{\partial u}{\partial x_k}\right)^{\mathrm{TSL}}_{e_{i,\mathrm{bdry}}} &= \begin{cases} \frac{-\left(n_{i,\mathrm{bdry}}\right)_k}{\left\|p_{i,\mathrm{bdry}} - p_{i,n}\right\|_2}\frac{1}{\rho_{i,n}}\begin{pmatrix} -u_{i,n,1} & 1 & 0 & 0 & 0 \\ -u_{i,n,2} & 0 & 1 & 0 & 0 \\ -u_{i,n,3} & 0 & 0 & 1 & 0 \end{pmatrix}, & m = i_n \\ 0, & m \neq i_n, \end{cases} \\
\frac{\partial}{\partial W_m}\left(\frac{\partial T}{\partial x_k}\right)^{\mathrm{TSL}}_{e_{i,\mathrm{bdry}}} &= 0, \qquad m = 1, \ldots, N_{\mathrm{elem}}.
\end{aligned}$$

Hence, following line by line the proof of Lemma 4.3.1 we get

$$\frac{\partial \left\langle \left(\overline{\mathcal{S}}\right)^{\mathrm{TSL}}_{e_{i,\mathrm{bdry}}}, n_{i,\mathrm{bdry}}\right\rangle}{\partial W_m} = 0, \qquad m \neq i_n$$

and

$$\begin{aligned}
&\frac{\partial \left\langle \left(\overline{\mathcal{S}}\right)^{\mathrm{TSL}}_{e_{i,\mathrm{bdry}}}, n_{i,\mathrm{bdry}} \right\rangle}{\partial W_{i_n}} \\
&= \frac{-1}{2\left\|p_{i,\mathrm{bdry}} - p_{i,n}\right\|_2 \rho_{i_n}} \begin{pmatrix} -N_1 u_{i,n,1} - N^{(i,n)}_{12,13} & N_1 & \frac{1}{3}N_{12} & \frac{1}{3}N_{13} & 0 \\ -N_2 u_{i,n,2} - N^{(i,n)}_{12,23} & \frac{1}{3}N_{12} & N_2 & \frac{1}{3}N_{23} & 0 \\ -N_3 u_{i,n,3} - N^{(i,n)}_{13,23} & \frac{1}{3}N_{13} & \frac{1}{3}N_{23} & N_3 & 0 \end{pmatrix},
\end{aligned}$$

where all terms need to be evaluated using the boundary normal $n_{i,\mathrm{bdry}}$. Due to the incorporation of the normal derivative of the temperature we obtain

$$\frac{\partial \left\langle (q)^{\mathrm{TSL}}_{e_{i,\mathrm{bdry}}}, n_{e_{i,\mathrm{bdry}}} \right\rangle}{\partial W_m} = 0, \qquad m = 1, \ldots, N_{\mathrm{elem}}.$$

Using the notation of Corollary 4.3.3, note that due to the choice (3.82a) the terms $E_1^{(i)}$ and $E_1^{(i,\mathrm{bdry})}$ vanish. To complete the derivative for the no-slip wall all these terms need to be incorporated into the terms presented in Section 4.3.

Symmetry boundary condition. The derivative of the symmetry boundary condition follows the considerations of the slip wall boundary condition. Here, one has to differentiate the same projection operator, which additionally needs to be combined with pre-computed values such as gradients, which requires an additional application the chain rule.

Chapter 5

Solution algorithms

This chapter is devoted to the design of a robust and reliable algorithm to solve the resulting large scale nonlinear systems of equations (3.97) arising from the discretization of the boundary value problems formulated in Section 2.4. Considering the remarks already made in the Introduction of this thesis, the challenges to approximate solutions to boundary value problems, for which the existence of solutions is an open problem, are manifold.

In particular, it is not in the range of expectations to find or to design a solution method which always works. One needs to be humble and honest, we do not expect that such a method exists in general, neither for linear nor for nonlinear problems, and in particular not for algebraic systems of nonlinear equations arising from a discretization of the RANS equations. Therefore, to be as flexible as possible, within this section it is the goal to derive a general solution methodology for nonlinear equations. Then, in a next step, we will concretize the general method exploiting certain features which come into play when considering the RANS equations, such that at least some major difficulties, which are inherent to the given problems, may be tackled. Moreover, one major theoretical result of this section is, that within the general framework presented here, almost all solution methods suggested in computational fluid dynamics can be identified as specializations choosing certain simplifications and parameters. An overview for the design of a solution method is given in Figure 5.1. Maybe the most important ingredient for a solution method is the inclusion of multigrid components.

Unfortunately but to no surprise, it turns out that almost all of the different ingredients we consider, depend on the choice of several parameters as well as truncation criteria. And, to be honest, for almost none of these parameters and truncation criteria we have a mathematical analysis in hands to determine those in an automatic self-adapting fashion.

Hence, being aware of the possibility that there does not exist a fixed, suited algorithm for nonlinear problems and a known parameter choice together with suitable

truncation criteria, it is a major goal of this work that several features of the algorithms can be easily arranged with respect to great flexibility. This idea is realized based upon a private implementation of a linear algebra package comprising many well known algorithms from numerical linear algebra. This package represents the backbone of the software which has been developed and used to compute the examples considered in Section 6. The design of this package is such, that in a straightforward way using parameter choice strategies, algorithms between full implicit and explicit methods can be selected. Start up strategies to find a good initial guess as well as stabilization strategies are included. This modular software design puts the solution algorithms into a position such that they can be easily adapted and extended for different problems and examples.

5.1 Solution methods for nonlinear equations

To approximately solve the algebraic system of equations (3.97) we apply a nonlinear multigrid method [96] called the Full Approximation Scheme (FAS). The approach in this thesis to realize multigrid components is based on the aggregation of degrees of freedom. This procedure is realized by the agglomeration of control volumes. It is the advantage of such a procedure that the coarse grid problem can be constructed directly from the finest grid level data. To this end, in a first step, the construction of coarse grid levels needs to be defined. Second, the formulation of the nonlinear multigrid together with projection and interpolation operators to transfer the data from one grid level to the next are required. And, finally, an effective smoother needs to be derived. Note that a robust and efficient nonlinear multigrid algorithm can only be expected in case all these components are synchronized to each other. One cannot expect a robust and reliable algorithm if these components are developed independently of each other. A general graphical overview to construct a powerful algorithm to solve a nonlinear operator equation is given in Figure 5.1. It shows the connection of several required ingredients. We start with the construction of coarse grid levels which are required for multigrid components.

5.1.1 Determination of lines

The presence of a boundary layer in convection-diffusion problems requires meshes with high aspect ratio cells along the no-slip boundary. They are used for the economic resolution of steep gradients. On the other hand, these cells are a major factor contributing to the loss in effectiveness of solution methods. One of the possible reasons is that these high aspect ratio cells result in a stiffness of the discrete system of governing flow equations. For analysis of the effects of such stiffness see Pierce and Giles [73].

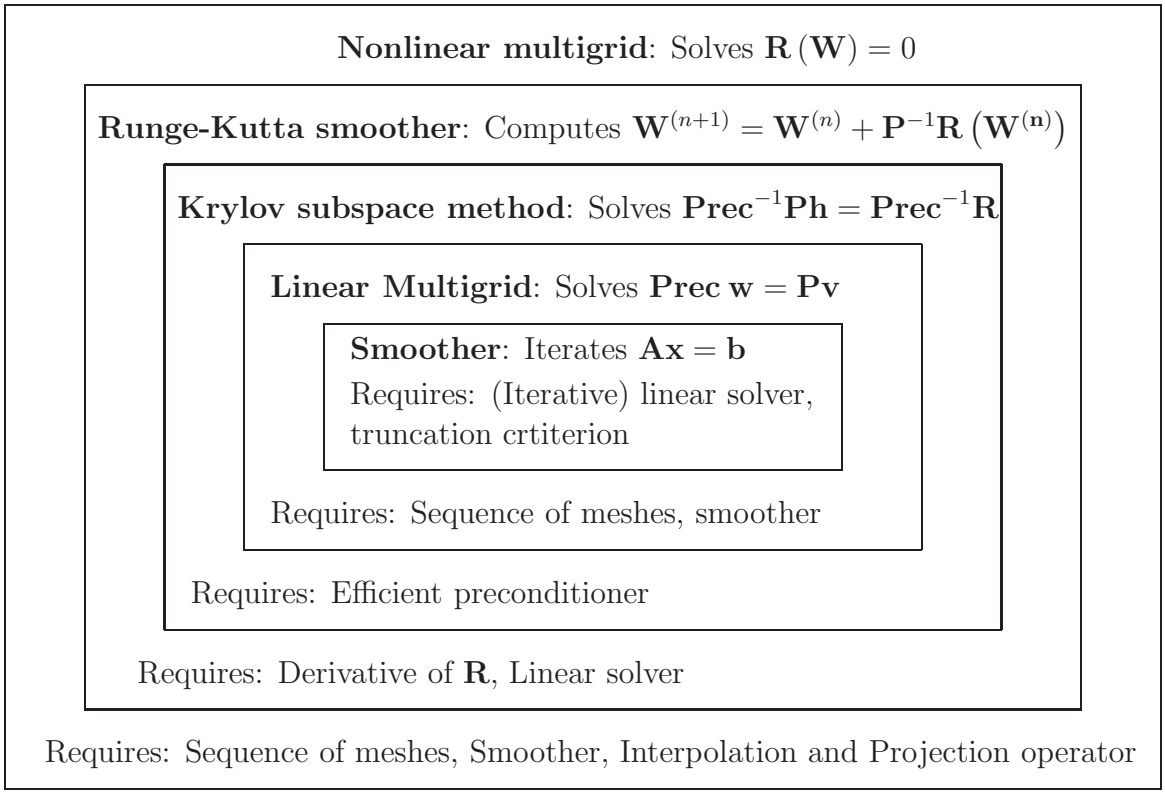

Figure 5.1: Algorithmical structure of nonlinear solution method

To deal with the severe anisotropies it is our goal to develop algorithmical techniques suited to identify sources of stiffness and to incorporate this knowledge appropriately into the solution algorithm. One key technology is the identification of elements in a given mesh representing an anisotropy. The aggregation of such elements and their corresponding degrees of freedom is what we call a line. An exact definition in our context is given below in Notation 5.1.2.

Grids given in an unstructured data format have no line information. This information must be generated. To this end we exploit, that in general meshes for high Reynolds number turbulent flows have a structured boundary layer with strong anisotropies. To detect these anisotropies, we formulate in this section a so-called line search algorithm. Similar line search algorithms have been suggested for example by Mavriplis [56] and Eliasson,Weinerfelt, and Nordström [16]. A line search algorithm based on the geometry of the boundary surface can be found in Nielsen et al. [66]. In the context of Discontinuous Galerkin methods, line search algorithms based on an advection-diffusion equation are established [19].

The line search algorithm presented here is based on a weighted graph method. To this end recall the definitions and notation of Section 3.1. To formulate the algorithm we introduce an edge weight based on the distance of the barycenters (3.1).

It is given by the inverse of the distance

$$w(e_{ij}) := (\mathrm{dist}(e_{ij}))^{-1}. \tag{5.1}$$

The ratio of maximum to average weight is used as an indication of the local anisotropy in the mesh at each vertex.

In the next step the vertices are sorted according to the ratio of the maximum to the average weight. This is an important issue since it ensures that lines originate in areas of maximum grid stretching and end in isotropic regions.

To construct the lines the first vertex in this ordered list is picked as the starting point for a line. The line is built by adding to the original vertex the neighboring vertex which is strongly connected to the current vertex, provided this vertex does not already belong to a line, and provided the ratio of maximum to minimum edge weights is greater than the threshold parameter $\gamma_{\mathrm{line}} \geq 1$. The line terminates when no additional vertex can be found.

Algorithm 5.1.1 *The line search algorithm can be summarized as follows:*

1) *For each vertex v_j, construct a list of edges $e_{ij}, i \in \mathcal{N}(j)$ originating from the vertex and determine the weight (5.1).*

2) *Compute the minimum weight, maximum weight, the average weight and the ratio of both of them:*

$$\begin{aligned} \mathrm{wmin}(v_j) &:= \min_{i\in\mathcal{N}(j)} \{w(e_{ij})\}, \qquad \mathrm{wmax}(v_j) := \max_{i\in\mathcal{N}(j)} \{w(e_{ij})\} \\ \mathrm{wavg}(v_j) &:= \frac{1}{\#\mathcal{N}(j)} \sum_{i=1}^{\#\mathcal{N}(j)} w(e_{ij}), \qquad \mathrm{rat}(v_j) := \mathrm{wmax}(v_j)/\mathrm{wavg}(v_j). \end{aligned}$$

3) *Sort the vertices v_j with respect to $\mathrm{rat}(v_j)$.*

4) *Construct the lines:*

   a) *Set searchOppositeDirection = false.*

   b) *Pick the first vertex v_k out of the sorted list, delete it from the list and add it to the line. Mark $\tilde{v} := v_k$.*

   c) *If the ratio $\mathrm{wmax}(v_k)/\mathrm{wmin}(v_k) \geq \gamma_{\mathrm{line}}$:*

      * *Find the neighbor vertex v_{neig} corresponding to $\mathrm{wmax}(v_k)$, delete it from the sorted list and add it to the line.*
      * *Define $v_k := v_{neig}$ and go back to c).*

   d) *else if the ratio $\mathrm{wmax}(v_k)/\mathrm{wmin}(v_k) < \gamma_{\mathrm{line}}$ and searchOppositeDirection = false:*

- *searchOppositeDirection = true*
- *Go back to the first element $\tilde{v}$ in the line (e.g. when a line starts in a wake region, one has to search in both directions from the original element). Set $v_k := \tilde{v}$.*
- *Go to c).*

e) else

- *Go to a).*

Notation 5.1.2 *Let M be a triangulation of D. We assume that n sets $L_1, \ldots, L_n$ of indices, $L_i = \{\ell_i^1, \ldots, \ell_i^{r_i}\}$, satisfying*

$$L_i \subset \Pi_G^M(M), \qquad L_j \cap L_i = \emptyset, \quad i \neq j, \qquad \cup_{j=1}^n L_j = \Pi_G^M(M), \quad r_i := \#(L_i),$$

are given. If $r_i > 1$, and the sequence of indices is a path in the graph $\left(\Pi_G^M(M), \partial\Pi_G^M(E(M))\right)$, that is

$$\left(\ell_i^1, \ell_i^2\right), \ldots, \left(\ell_i^{r_i-1}, \ell_i^{r_i}\right), \qquad \left(\ell_i^{j-1}, \ell_i^j\right) \in \partial\Pi_G^M(E(M)), \quad j = 2, \ldots, r_i$$

such that the indices $\ell_i^1, \ldots, \ell_i^{r_i}$ are pairwise distinct, we call the set L_i a line. Otherwise we say L_i is a point.

Algorithm 5.1.1 determines n sets of points satisfying the postulated properties used in Notation 5.1.2.

Note that Algorithm 5.1.1 is only one possibility to construct sets of indices satisfying the properties postulated in Notation 5.1.2. For example, in case the original grid is structured and the indices of the no-slip boundary are explicitly known, then one can directly identify lines which originate on the no-slip boundary and propagate as a ray into the farfield. The actual implementation of the finding of the sets of points representing strong anisotropies is not that important. It is important to identify sets of points representing directions of strong coupling and to include this information into the solution algorithm. We will exploit these directions of strongest coupling twice. On the one hand, our agglomeration technique and the construction of coarse multigrid levels is based on it. This is topic of the next Section 5.1.2. And second, we will incorporate the information into the smoother used in the nonlinear multigrid, described in Section 5.2.3.

5.1.2 Agglomeration techniques

The formulation of multigrid algorithms requires the formulation of coarse grid and fine grid equations. Therefore, we give the the following definition.

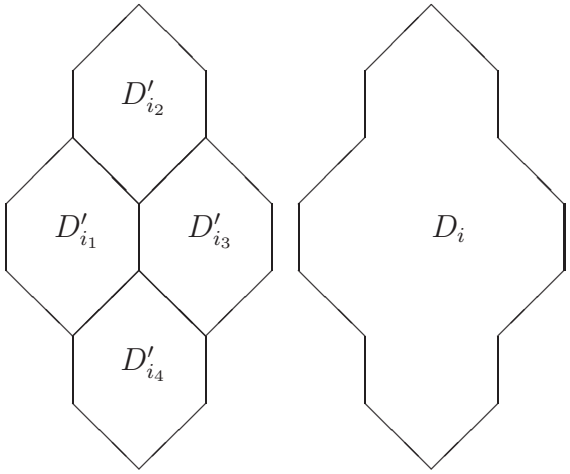

Figure 5.2: Four cells of a dual grid (left) and their agglomerated cell (right)

Definition 5.1.3 *Assume that*

$$M = \{D_i : i = 1, \dots N_{elem}\} \qquad \text{and} \qquad M' = \{D'_i : i = 1, \dots N'_{elem}\}$$

are triangulations of the bounded domain $D \subset \mathbb{R}^m$. M' is called refinement of M if for all $D_i \in M$, $i = 1, \dots, N_{elem}$ there exists a set of indices

$$C(i) := \{i_1, \dots, i_N\}$$

such that

$$J_i = \left\{D'_{i_1}, \dots, D'_{i_N}\right\}$$

is a triangulation of D_i. We then write $M \subset M'$. An element $k \in C(i)$ is called a child of D_i. Synonymously, we also denote the corresponding domain D'_k, $k \in C(i)$, as child of D_i.

An graphical example for such a refinement is given in Figure 5.2. Here we have

$$C(i) = \{i_1, i_2, i_3, i_4\}$$

and the children of D_i are given by D'_{i_j}, $j = 1, \dots, 4$.

To formulate a multigrid algorithm we need to construct a sequence of triangulations

$$M_n \subset \dots \subset M_1.$$

To do so we apply agglomeration techniques based on weighted graph algorithms implemented in the library MGridGen [64]. To make this method applicable for large scale applications with meshes with extensive anisotropic cells it was extended

by a directional agglomeration strategy [55, 56, 49]. Regions of high grid stretching are identified using Algorithm 5.1.1.

Along a line, a predetermined number of points is fused to one coarse cell. To reduce significantly the mesh induced stiffness on the agglomerated meshes, two points are usually fused. This procedure results in a sequence of coarse grid levels for which the complexity between successive levels decreases by a factor of 2. In regions of the mesh where no line information is available – usually the isotropic part of the mesh – MGridGen is applied yielding approximately a $4:1$ in 2D and $8:1$ in 3D fusing of the cells.

Algorithm 5.1.4 *The coarsening algorithm may be written in pseudo code as follows:*

- *Search and construct lines on finest grid level using Algorithm 5.1.1*
- *Construct a new pseudo mesh in which a line is represented by a point*
- *Construct required data of pseudo mesh for MGridGen*
- *Agglomerate pseudo mesh using MGridGen*
- *Unpack the lines and agglomerate by the relation* $2:1$

Remark 5.1.5 *Since the coarsening algorithm fuses fine grid cells of* M' *to a coarse grid cell of* M*, it is trivial that* $M \subset M'$ *is satisfied.*

Note that we combine two kinds of coarsening strategies. On the one hand there is a isotropic far field within the mesh, and on the other hand we have an anisotropic part near the no-slip wall. So, for coarsening the far field region we use the graph coarsening algorithm MGridGen. But this tool is not allowed to touch the anisotropic part. Hence, in a first step a pseudo mesh needs to be constructed which is given to MGridGen. To this end the predetermined lines are in the pseudo mesh represented as a point. Then this mesh is coarsened by MGridGen. Then the points representing a line are unpacked in the coarse mesh. These are not yet coarsened. However, along a line one can simply fuse two neighboring cells. So, the coarsening ratio along the lines is 2:1, in the rest of the field it is given by the coarsening ratio of the graph coarsening tool.

It should be noted that for agglomerating the isotropic part of the mesh any other method than MGridGen may be used, such as an advancing front algorithm. The important part is a special treatment of the anisotropic part of the mesh.

Figures 5.3 and 5.4 present four coarse grid levels in the boundary layer of an airfoil. The red lines represent the original fine mesh. The thick blue lines form the agglomerated meshes. It can be seen that the methodology explained previously yields exactly a directional $2:1$ coarsening in the anisotropic part of the mesh. Coarsening examples for the isotropic part of the mesh are given in Figures 5.5 – 5.8.

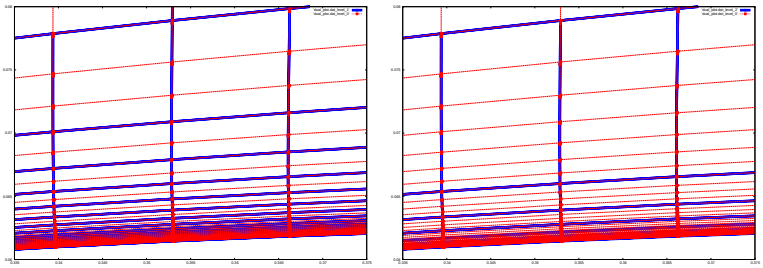

Figure 5.3: Directional coarsening strategy in anisotropic section of the mesh, left: 1st agglomerated mesh, right: 2nd agglomerated mesh

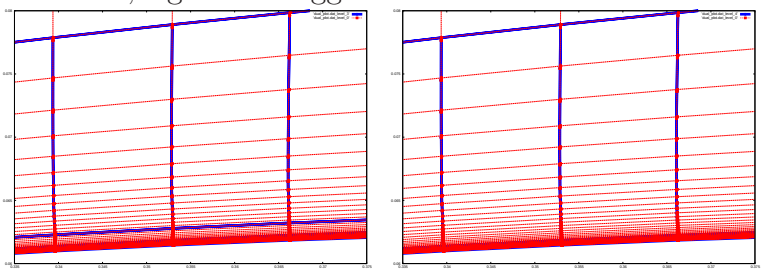

Figure 5.4: Directional coarsening strategy in anisotropic section of the mesh, left: 3rd agglomerated mesh, right: 4th agglomerated mesh

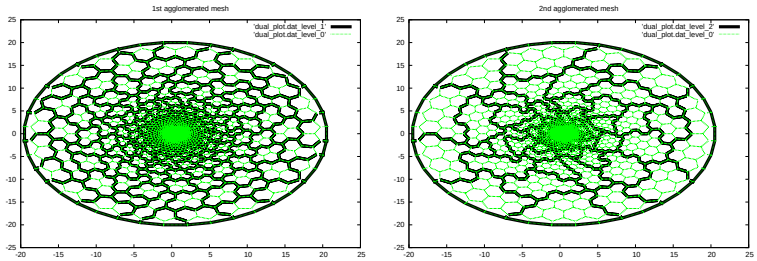

Figure 5.5: Standard coarsening strategy in the isotropic section of the mesh in the farfield, left: 1st agglomerated mesh, right: 2nd agglomerated mesh

5.1.3 Nonlinear multigrid

For a detailed description of multigrid methods we refer to the textbook of Trottenberg et al. [96]. The nonlinear multigrid method considered in this thesis is based on the aggregation of degrees of freedom. This procedure is realized by the agglomeration of control volumes. The formulation of the nonlinear multigrid together requires projection and interpolation operators to transfer the data from one grid level to the next. And, finally, an effective smoother needs to be derived.

The fundamental idea of nonlinear multigrid is to smooth the errors such that these can be represented on coarser grids. Then, the errors on the corresponding coarse grid equation are smoothed and the coarse grid corrections are interpolated back to

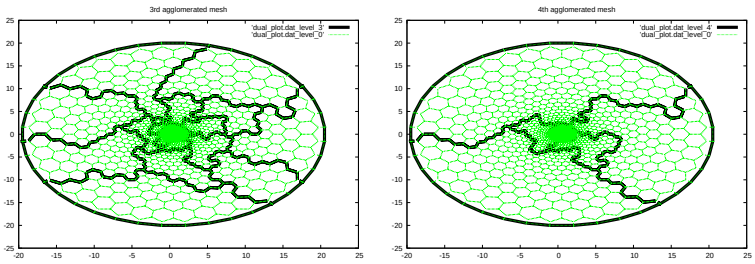

Figure 5.6: Standard coarsening strategy in the isotropic section of the mesh in the farfield, left: 3rd agglomerated mesh, right: 4th agglomerated mesh

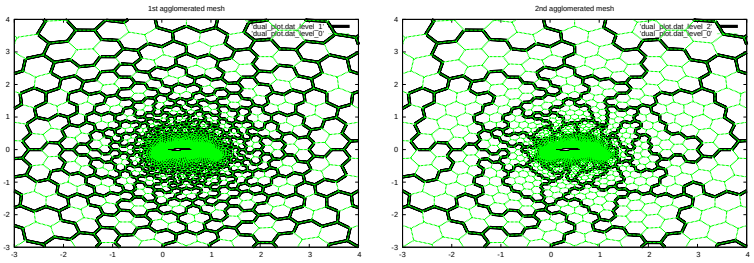

Figure 5.7: Standard coarsening strategy in the isotropic section of the mesh near the airfoil, left: 1st agglomerated mesh, right: 2nd agglomerated mesh

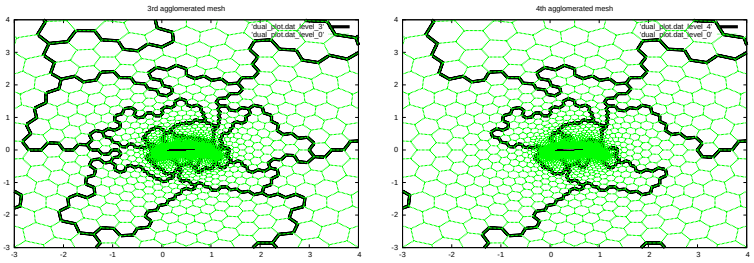

Figure 5.8: Standard coarsening strategy in the isotropic section of the mesh near the airfoil, left: 3rd agglomerated mesh, right: 4th agglomerated mesh

the fine grid. To formulate the nonlinear Full Approximation Scheme we consider the nonlinear equation

$$\mathbf{R}_{M_1} : \mathbb{R}^{\#M_1} \to \mathbb{R}^{\#M_1}, \qquad \mathbf{R}_{M_1}(\mathbf{W}_{M_1}) = \mathbf{f}_{M_1}, \tag{5.2}$$

and define for $2 \leq k \leq n$ a projection from a fine to the next coarser mesh and an interpolation from a coarse to the next finer mesh:

$$\begin{aligned} P_{M_k}^{M_{k-1}} &: \mathbb{R}^{\#M_{k-1}} \to \mathbb{R}^{\#M_k}, \\ I_{M_k}^{M_{k-1}} &: \mathbb{R}^{\#M_k} \to \mathbb{R}^{\#M_{k-1}}. \end{aligned}$$

The operator $\mathbf{R}_{M_1}$ represents for example the discretization of partial differential equation or of an integral equation on a given mesh. In case of the integral

equation (2.1a) or (2.45) resp. $\mathbf{R}_{M_1}$ represents furthermore a system of equations with $d > 1$ degrees of freedom (see (3.10)) per control volume. Then the operator equation (5.2) is replaced by

$$\mathbf{R}_{M_1} : \mathbb{R}^{d\#M_1} \to \mathbb{R}^{d\#M_1}, \qquad \mathbf{R}_{M_1}(\mathbf{W}_{M_1}) = \mathbf{f}_{M_1}. \tag{5.3}$$

Assume that the solution $\mathbf{W}^*_{M_1}$ of (5.2) in the step ℓ of the iteration can be represented by the correction $\mathbf{V}^{(\ell)}_{M_1}$ and a given approximate solution $\mathbf{U}^{(\ell)}_{M_1}$,

$$\mathbf{W}^*_{M_1} = \mathbf{U}^{(\ell)}_{M_1} + \mathbf{V}^{(\ell)}_{M_1}.$$

Then we obtain the defect equation on the grid M_1 given by

$$\mathbf{R}_{M_1}\left(\mathbf{U}^{(\ell)}_{M_1} + \mathbf{V}^{(\ell)}_{M_1}\right) - \mathbf{R}_{M_1}\left(\mathbf{U}^{(\ell)}_{M_1}\right) = \mathbf{d}^{(\ell)}_{M_1}, \tag{5.4}$$

which needs to be solved for $\mathbf{U}^{(\ell)}_{M_1}$. Note that if $\mathbf{U}^{(\ell)}_{M_1} = \mathbf{W}^*_{M_1}$ is the solution of (5.3) we conclude that the correction and the defect satisfy $\mathbf{V}^{(\ell)}_{M_1} = 0$ and $\mathbf{d}^{(\ell)}_{M_1} = 0$.
An approximation of this equation on the next coarse grid M_2 is

$$\mathbf{R}_{M_2}\left(\mathbf{U}^{(\ell)}_{M_2} + \mathbf{V}^{(\ell)}_{M_2}\right) - \mathbf{R}_{M_2}\left(\mathbf{U}^{(\ell)}_{M_2}\right) = \mathbf{d}^{(\ell)}_{M_2}, \tag{5.5}$$

where $\mathbf{R}_{M_2}$ represents an approximation to the nonlinear operator $\mathbf{R}_{M_1}$ and the other missing terms are given by an application of the projection operator,

$$\begin{aligned}
\mathbf{U}^{(\ell)}_{M_2} &:= P^{M_1}_{M_2}\mathbf{U}^{(\ell)}_{M_1},\\
\mathbf{V}^{(\ell)}_{M_2} &:= P^{M_1}_{M_2}\mathbf{V}^{(\ell)}_{M_1},\\
\mathbf{d}^{(\ell)}_{M_2} &:= P^{M_1}_{M_2}\mathbf{d}^{(\ell)}_{M_1}.
\end{aligned}$$

Now, the equation (5.5) can be solved for $\mathbf{V}^{(\ell)}_{M_2}$ such that

$$\mathbf{R}_{M_2}\left(\mathbf{U}^{(\ell)}_{M_2} + \mathbf{V}^{(\ell),*}_{M_2}\right) = \mathbf{d}^{(\ell)}_{M_2} + \mathbf{R}_{M_2}\left(\mathbf{U}^{(\ell)}_{M_2}\right), \tag{5.6}$$

to obtain for the defect equation (5.4) the correction

$$\mathbf{U}^{(\ell+1)}_{M_1} := \mathbf{U}^{(\ell)}_{M_1} + I^{M_1}_{M_2}\mathbf{V}^{(\ell),*}_{M_2}. \tag{5.7}$$

In general the coarse grid defect equation (5.5) is not solved exactly but only approximately. The nonlinear multigrid scheme is summarized in the following algorithm.

Algorithm 5.1.6 *Assume that for a sequence of meshes $M_n \subset \ldots \subset M_1$ the operator equations*

$$\mathbf{R}_{M_k}\left(\mathbf{U}^{(\ell)}_{M_k}\right) = \mathbf{f}_{M_k}, \qquad k = 1, \ldots, n,$$

and an approximate solution $\mathbf{U}_{M_k}^{(\ell)}$ *together with a smoother* $\mathbf{S} = \mathbf{S}\left(\mathbf{U}_{M_k}^{(\ell)}, \mathbf{R}_{M_k}, \mathbf{f}_{M_k}\right)$ *are given. Here* $\mathbf{S}$ *denotes a method which determines from the approximate solution* $\mathbf{U}_{M_k}^{(\ell)}$ *an update*

$$\overline{\mathbf{U}}_{M_k}^{(\ell)} := \mathbf{S}\left(\mathbf{U}_{M_k}^{(\ell)}, \mathbf{R}_{M_k}, \mathbf{f}_{M_k}\right).$$

Let us denote by

- $\omega_{M_k,\mathrm{pre}} \in \mathbb{N}_0$ *the number of presmoothing steps*
- $\omega_{M_{k+1}} \in \mathbb{N}_0$ *the number of coarse grid smoothing steps*
- $\omega_{M_k,\mathrm{post}} \in \mathbb{N}_0$ *the number of postsmoothing steps*

Then we define a $2V = 2V\left(k, \omega_{M_k,\mathrm{pre}}, \omega_{M_{k+1}}, \omega_{M_k,\mathrm{post}}\right)$ *multigrid cycle by*

a) Apply $\omega_{M_k,\mathrm{pre}}$ *smoothing steps, that is compute*

$$\overline{\mathbf{U}}_{M_k}^{(\ell)} = \mathbf{S} \circ \cdots \circ \mathbf{S}\left(\mathbf{U}_{M_k}^{(\ell)}, \mathbf{R}_{M_k}, \mathbf{f}_{M_k}\right) = \mathbf{S}^{\omega_{M_k,\mathrm{pre}}}\left(\mathbf{U}_{M_k}^{(\ell)}, \mathbf{R}_{M_k}, \mathbf{f}_{M_k}\right).$$

b) Compute the right hand side of the coarse grid equation

$$\mathbf{f}_{M_{k+1}} = P_{M_{k+1}}^{M_k}\left(\mathbf{f}_{M_k} - \mathbf{R}_{M_k}\left(\overline{\mathbf{U}}_{M_k}^{(\ell)}\right)\right) + \mathbf{R}_{M_{k+1}}\left(P_{M_{k+1}}^{M_k}\overline{\mathbf{U}}_{M_k}^{(\ell)}\right).$$

c) Apply $\omega_{M_{k+1}}$ *coarse grid smoothing steps, that is compute*

$$\overline{\mathbf{U}}_{M_{k+1}}^{(\ell)} = \mathbf{S}^{\omega_{M_{k+1}}}\left(P_{M_{k+1}}^{M_k}\overline{\mathbf{U}}_{M_k}^{(\ell)}, \mathbf{R}_{M_{k+1}}, \mathbf{f}_{M_{k+1}}\right).$$

d) Compute an update of the approximate solution

$$\overline{\mathbf{U}}_{M_k}^{(\ell+1)} := \overline{\mathbf{U}}_{M_k}^{(\ell)} + I_{M_{k+1}}^{M_k}\left(\overline{\mathbf{U}}_{M_{k+1}}^{(\ell)} - P_{M_{k+1}}^{M_k}\overline{\mathbf{U}}_{M_k}^{(\ell)}\right).$$

e) Apply $\omega_{M_k,\mathrm{post}}$ *smoothing steps, that is compute*

$$\mathbf{U}_{M_k}^{(\ell+1)} = \mathbf{S}^{\omega_{M_k,\mathrm{post}}}\left(\overline{\mathbf{U}}_{M_k}^{(\ell+1)}, \mathbf{R}_{M_k}, \mathbf{f}_{M_k}\right).$$

The $2V$ multigrid cycle formulated in Algorithm 5.1.6 may be viewed as a prototype to formulate multigrid cycles in general. Famous cycling strategies are so called V-cycles, W-cycles and F-cycles. These kind of cycles can either be defined explictly or recursively.

The nonlinear multigrid described by Algorithm 5.1.6 requires several components. First of all, it needs a sequence of meshes $M_n \subset \ldots \subset M_1$. The generation of such a sequence was described in Section 5.1.2. Second, the definition of suited restriction and interpolation operators are required, which is topic of Section 5.1.4. And finally, a problem appropriate smoother needs to be constructed. This is topic of Section 5.1.6.

5.1.4 Construction of transfer operators

To derive projection $P_{M_{k+1}}^{M_k}$ and interpolation $I_{M_{k+1}}^{M_k}$ operators we need the definition of the ansatz space (3.4), together with the L^2 scalar product for the domain D,

$$(f,g)_{L^2(D)} = \int_D f(x)g(x)dx.$$

Exploiting a triangulation M of D, we obtain for two functions $f, g \in Sp_0(M)$ the representation

$$(f,g)_{L^2(D)} = (f,g)_{L^2(M)} = \sum_{j=1}^{N_{elem}} \int_{D_j} f_j \mathbb{1}_{D_j}(x) g_j \mathbb{1}_{D_j}(x)dx = \sum_{j=1}^{N_{elem}} \mathrm{vol}(D_j)\, f_j g_j.$$

Since an orthonormal basis of $Sp_0(M)$ is given by

$$\left\{ \frac{1}{\sqrt{\mathrm{vol}(D_1)}} \mathbb{1}_{D_1}, \ldots, \frac{1}{\sqrt{\mathrm{vol}(D_{N_{elem}})}} \mathbb{1}_{D_{N_{elem}}} \right\},$$

and assuming that M' is a refinement of M, we define a projection

$$P \quad : \quad Sp_0(M') \to Sp_0(M), \tag{5.8a}$$

$$Pf \quad := \quad \sum_{i=1}^{N_{elem}} \left(f, \frac{1}{\sqrt{\mathrm{vol}(D_i)}} \mathbb{1}_{D_i} \right)_{L^2(M)} \frac{1}{\sqrt{\mathrm{vol}(D_i)}} \mathbb{1}_{D_i}. \tag{5.8b}$$

Hence, corresponding to the ansatz function (3.9) representing $f \in Sp_0(M')$ by

$$f = \sum_{j=1}^{N'_{elem}} f_j \mathbb{1}_{D'_i},$$

the projection is computed explicitly by

$$\begin{aligned} Pf &= \sum_{i=1}^{N_{elem}} \left(\sum_{j=1}^{N'_{elem}} f_j \mathbb{1}_{D'_j}, \frac{1}{\sqrt{\mathrm{vol}(D_i)}} \mathbb{1}_{D_i} \right)_{L^2(M)} \frac{1}{\sqrt{\mathrm{vol}(D_i)}} \mathbb{1}_{D_i} \\ &= \sum_{i=1}^{N_{elem}} \frac{1}{\mathrm{vol}(D_i)} \left(\sum_{j=1}^{N'_{elem}} f_j \mathbb{1}_{D'_j}, \mathbb{1}_{D_i} \right)_{L^2(M)} \mathbb{1}_{D_i} \\ &= \sum_{i=1}^{N_{elem}} \frac{1}{\mathrm{vol}(D_i)} \left(\sum_{j \in C(i)} \mathrm{vol}\left(D'_j\right) f_j \right) \mathbb{1}_{D_i}. \end{aligned} \tag{5.9}$$

This consideration suggests to define the the projection operator for Algorithm 5.1.6 by the linear mapping

$$P_{M_k}^{M_{k-1}} := \left(\frac{1}{\mathrm{vol}\,(D_i)} \left(\mathrm{vol}\left(D'_{i_1}\right), \ldots, \mathrm{vol}\left(D'_{i_N}\right)\right)\right)_{i=1,\ldots,N_{elem}}. \tag{5.10}$$

Corresponding to the projection (5.8) we define the interpolation

$$I \quad : \quad Sp_0(M) \to Sp_0(M'), \tag{5.11a}$$

$$Ig \quad := \quad \sum_{i=1}^{N'_{elem}} \left(g, \frac{1}{\sqrt{\mathrm{vol}\,(D'_i)}} \mathbb{1}_{D'_i} \right)_{L^2(M')} \frac{1}{\sqrt{\mathrm{vol}\,(D'_i)}} \mathbb{1}_{D'_i}. \tag{5.11b}$$

Then, for a function $g \in Sp_0(M)$ represented by $g = \sum_{j=1}^{N_{elem}} g_j \mathbb{1}_{D_j}$ we compute

$$\begin{aligned}
Ig &= \sum_{i=1}^{N'_{elem}} \frac{1}{\mathrm{vol}\,(D'_i)} \left(\sum_{j=1}^{N_{elem}} g_j \mathbb{1}_{D_j}, \mathbb{1}_{D'_i} \right)_{L^2(M')} \mathbb{1}_{D'_i} \\
&= \sum_{i=1}^{N'_{elem}} \frac{1}{\mathrm{vol}\,(D'_i)} \left(\sum_{j=1}^{N_{elem}} g_j \sum_{k=1}^{N'_{elem}} \int_{D'_k} \mathbb{1}_{D_j}(x) \mathbb{1}_{D'_i}(x) dx \right) \mathbb{1}_{D'_i} \\
&= \sum_{j=1}^{N_{elem}} g_j \left(\sum_{i=1}^{N'_{elem}} \frac{1}{\mathrm{vol}\,(D'_i)} \int_{D_j \cap D'_i} \mathbb{1}_{D_j}(x) \mathbb{1}_{D'_i}(x) dx \right) \mathbb{1}_{D'_i} \\
&= \sum_{i=1}^{N_{elem}} g_{C(i)} \mathbb{1}_{D'_{C(i)}}.
\end{aligned}$$

This consideration suggests to define the the interpolation operator for Algorithm 5.1.6 by the linear mapping

$$I_{M_k}^{M_{k+1}} := (1_{i_1}, \ldots, 1_{i_N})_{i=1,\ldots,N_{elem}}^T. \tag{5.12}$$

By a direct computation it follows that the projection P and the interpolation I are adjoint,

$$\begin{aligned}
(Pf, g)_{L^2(M)} &= \sum_{j=1}^{N_{elem}} \mathrm{vol}\,(D_j)\,(Pf)_j\, g_j \\
&= \sum_{j=1}^{N_{elem}} \mathrm{vol}\,(D_j) \left(\frac{1}{\mathrm{vol}\,(D_j)} \sum_{i \in C(j)} \mathrm{vol}\,(D'_i)\, f_i \right) g_j \\
&= \sum_{i=1}^{N'_{elem}} \mathrm{vol}\,(D'_i)\, f_i g_{C(i)} = (f, Ig)_{L^2(M')}.
\end{aligned}$$

It is important to notice that this understanding of projection and interpolation allows to generalize these operators directly for the general ansatz space Sp_k defined in (3.4), a requirement to formulate multigrid for discontinuous Galerkin methods. Naturally, then for all ansatz functions the integrals over the subdomains need to integrated which means a significant increase in complexity. Nevertheless, this can be done in a preprocessing step.

Remark 5.1.7 *In a discrete setting both the projection operator (5.10) and the interpolation operator (5.12) can be represented using sparse matrices.*

For the formulation of Algorithm 5.1.6 we have not distinguished between the interpolation and projection operators in the image and pre-image space. To formulate item b), we still need a projection in the image space. Therefore, we use the representation

$$\mathbf{R}(\mathbf{W}) = \sum_{i=1}^{N'_{elem}} \frac{1}{\mathrm{vol}(D'_i)} (\mathbf{R})_i \, \mathbb{1}_{D'_i}.$$

Then we compute similar to (5.9)

$$P\mathbf{R}(\mathbf{W}) = \sum_{i=1}^{N_{elem}} \frac{1}{\mathrm{vol}(D_i)} \left(\sum_{j \in C(i)} \mathbf{R}_j \right) \mathbb{1}_{D_i}. \tag{5.13}$$

Hence, (5.10) can be used to project the residual $\mathbf{R}$. In the case of general ansatz spaces Sp_k the inverse of the volumes need to be replaced by the inverse of the mass matrix.

5.1.5 Coarse grid equations

Coarse grid equations are assembled using an aggregation method. We explain our procedure by considering Figure 5.2, representing an example of a typical 2D dual grid cell arising from a triangular mesh and its agglomerated coarse grid cell. We use the notation of Definition 5.1.3. Assume that $D_i^{(k)}$ is the ith coarse grid cell on grid level k, which was created by fusing $D_{i_1}^{(k-1)}, \ldots, D_{i_N}^{(k-1)}$, that is

$$D_{i_j}^{(k-1)} \subset D_i^{(k)}, \qquad \bigcup_{j \in C^{(k-1)}(i)} \overline{D}_{i_j}^{(k-1)} = \overline{D}_i^{(k)}.$$

To keep up with the idea of aggregation of degrees of freedom in correspondence with the ansatz space $Sp_0(M)$ defined in (3.4), we need to translate this to coarse grids. This is required to compute the residual on the kth agglomerated mesh.

Using the interpolation operator (5.12), in a first step the coarse grid flow variables $\mathbf{W}_i^{(k)}$ corresponding to grid level k and cell i are prolongated to the finest grid level,

$$\mathbf{W}_j^{(1)} := \mathbf{W}_i^{(k)}, \qquad j \in C^{(1)}(i). \tag{5.14}$$

The interpolation operator required for (5.14) can either be implemented directly using (5.12). This is possible because of constructing coarse grid levels by agglomeration. Otherwise, the operation required for (5.14) can be implemented recursively,

$$\mathbf{W}_j^{(\ell-1)} := \mathbf{W}_i^{(\ell)}, \qquad j \in C^{(k-1)}(i), \quad \ell = k, \ldots, 2.$$

These operations can either be implemented directly, or, as mentioned in Section 5.1.4, using for example sparse matrices given by (5.12),

$$\mathbf{W}^{(1)} = I_{M_2}^{M_1} \circ \ldots \circ I_{M_k}^{M_{k-1}} \mathbf{W}^{(k)}.$$

Considering generalizations to function spaces $Sp_k(M)$ usage of sparse matrices might be preferred.

Then the fluxes for all edges $E(M_1)$ and $E_{\text{bdry}}(M_1)$ for the interpolated variables $\mathbf{W}^{(1)}$ to the finest grid level M_1 are computed. The coarse grid residual corresponding to control volume $D_i^{(k)}$ on grid level k is evaluated by summing up all the fluxes on the edges corresponding to the fine grid cells representing the coarse grid cell,

$$\mathbf{R}_i^{(k)}\left(\mathbf{W}^{(k)}\right) = \sum_{j \in C^{(1)}(i)} \sum_{e \in E(D_{i_j}^{(1)})} (\mathbf{f}_e \cdot n_e),$$

where $\mathbf{f}_e$ denotes the flux and n_e the normalized normal vector for edge e. Since all flux terms corresponding to inner edges for cell $D_i^{(k)}$ cancel out, this construction yields,

$$\mathbf{R}_i^{(k)}\left(\mathbf{W}^{(k)}\right) = \sum_{e \in E(D_i^{(k)})} (\mathbf{f}_e \cdot n_e). \tag{5.15}$$

Therefore, formally only the outer fluxes on the coarse grid cells need to be evaluated and summed up. Such an assembling strategy of coarse grid equations is advantageous when compared to a discretization strategy (see e.g. [20]). In particular, no problems with fusing boundary conditions and introducing a geometry on the agglomerates are encountered. Moreover, the geometry of the finest grid level M_1 is simply reused as well as the evaluations of the fluxes.

The coarse grid equation (5.15) comprises all terms, which includes convective, viscous and for the turbulent flow equation also source terms. The convective terms on coarse grids are evaluated using the flux given in Definition 3.2.4. Gradients (3.58) required for the viscous and source terms are computed following the same aggregation idea. Being aware of the fact that this procedure may lead to an inconsistency with respect to viscous terms (see [54, 67, 95]) the viscous terms are weighted with a factor of $(1/2)^{k-1}$. This factor is then also consequently used in the construction of the derivative terms used in the coarse grid preconditioner and the coarse grid ΔT. Finally, the projection operator (5.10) can be implemented either directly

$$\mathbf{W}_i^{(k)} = \frac{1}{\text{vol}\left(D_i^{(k)}\right)} \sum_{j \in C^{(k)}(i)} \text{vol}\left(D_{i_j}^{(1)}\right) \mathbf{W}_j^{(1)},$$

or recursively,

$$\mathbf{W}_i^{(\ell)} = \frac{1}{\text{vol}\left(D_i^{(\ell)}\right)} \sum_{j \in C^{(\ell)}(i)} \text{vol}\left(D_{i_j}^{(\ell-1)}\right) \mathbf{W}_j^{(\ell-1)}, \qquad \ell = 2, \ldots, k.$$

Again, this can be globally expressed using (5.10),

$$\mathbf{W}^{(k)} = P_{M_k}^{M_{k-1}} \circ \ldots \circ P_{M_2}^{M_1} \mathbf{W}^{(1)}.$$

For the implementation of the projection operator (5.13) note that within our implementation we have not included $\mathbf{M}^{-1}$ in the definition of the residual $\mathbf{R}$. Hence, it is implemented

$$\mathbf{R}_i^{(k)} = \sum_{j \in C^{(k)}(i)} \mathbf{R}_j^{(1)}. \tag{5.16}$$

Comparing (5.13) and (5.16) we note that multiplication with $\mathbf{M}^{-1}$ is therefore missing for all grid levels. This missing operation is corrected by considering in the next Section 5.1.6 by explicitly including $\mathbf{M}^{-1}$. Note that this is not necessary in case one defines the residual $\mathbf{R}$ by including $\mathbf{M}^{-1}$ into the definition.
To compute the time step size matrix ΔT and the preconditioner $\mathbf{P}_j$ in Algorithm (5.20) on a coarse grid level, exactly the same procedure as in (5.15) is applied. Since the convective terms of (2.1a) on coarse grid levels are only approximated by a first order Roe scheme (3.18), we have agreement with respect to the order of approximation in the residual function and the preconditioner. In deviations for the preconditioner, the Roe matrix and both the laminar and eddy viscosities are assumed to be constant [42, 47].

Using this multigrid approach, in particular, boundary conditions on the coarse grid levels can be implemented in a straightforward manner. They simply arise from the boundary conditions given on the finest mesh. Therefore, the particular problems of fusing boundary cells and different boundary conditions is completely avoided. The multigrid operators described above are then included into a FAS scheme, The construction of a smoother is topic of Section 5.1.6.

In all the numerical examples considered in Chapter 6 we apply standard cycling strategies 2v, 3v, and 4w as described in [96]. Dealing with nonlinear problems we do not solve on the coarsest grid level, but we only perform a smoothing step.

5.1.6 Construction of a multigrid smoother

To derive a suitable smoother for Algorithm 5.1.6 to solve the discretized flow equations (3.97a) and (3.97b), we consider the time dependent equations (3.96a) and (3.96b). As already mentioned in the introduction, to overcome the stiffness of the problems considered in this thesis a powerful method having the capability to smooth a variety of error components is required. Hence, we do not follow the idea to

combine multigrid with a smoother of low operational cost. Instead, contradicting the idea of multigrid, we derive a smoother based on multi stage implicit Runge-Kutta methods. Therefore, consider an s-stage diagonally implicit Runge-Kutta method given by the Butcher scheme

$$\begin{array}{c|c} c & \mathcal{A} \\ \hline & b^T \end{array}$$

Table 5.1: Butcher scheme

where

$$\mathcal{A} := \begin{pmatrix} \alpha_{11} & 0 & \cdots & 0 \\ \alpha_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \alpha_{s,s-1} & \alpha_{ss} \end{pmatrix}, \quad b := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha_{s+1,s} \end{pmatrix} \quad \text{and} \quad c := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{5.17}$$

In the following we skip the sub-indices "mean" and "turb" since we use the same method for solving (3.96a) and (3.96b). Then, denoting the discrete evolution at T_n by $\mathbf{W}^{\mathrm{T}_n}$, an application of this Butcher scheme to (3.96a) and (3.96b) gives the stages and next discrete evolution

$$\begin{aligned} k_1 &= -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n} + \alpha_{11}\Delta t k_1\right) \\ k_2 &= -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n} + \alpha_{21}\Delta t k_1 + \alpha_{22}\Delta t k_2\right) \\ &\vdots \\ k_s &= -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n} + \alpha_{s,s-1}\Delta t k_{s-1} + \alpha_{ss}\Delta t k_s\right) \\ \mathbf{W}^{\mathrm{T}_{n+1}} &= \mathbf{W}^{\mathrm{T}_n} + \alpha_{s+1,s}\Delta t k_s. \end{aligned} \tag{5.18}$$

To approximate a solution of the nonlinear systems $k_1, \ldots, k_s$ we use Newton's method truncated after only one iteration to approximate the root of the function

$$g_j(k) := k + \mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n} + \alpha_{j,j-1}\Delta t k_{j-1} + \alpha_{jj}\Delta t k\right)$$

Its derivative is given by

$$\frac{dg_j(k)}{dk} = \mathbf{I} + \alpha_{jj}\Delta t \mathbf{M}^{-1}\frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^{\mathrm{T}_n} + \alpha_{j,j-1}\Delta t k_{j-1} + \alpha_{jj}\Delta t k\right),$$

and the initial guess is assumed to be $k^{(0)} = 0$. Then an approximate root for stages $j = 1, \ldots, s$ is given by

$$k_j = -\left[\frac{dg_j\left(k^{(0)}\right)}{dk}\right]^{-1}\left(g_j\left(k^{(0)}\right)\right). \tag{5.19}$$

Using the approximate root (5.19), the implicit Runge-Kutta method (5.18), is represented by the algorithm

$$\begin{aligned}
k_1 &= -\left[\frac{dg_1\left(k^{(0)}\right)}{dk}\right]^{-1}\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n}\right)\\
k_2 &= -\left[\frac{dg_2\left(k^{(0)}\right)}{dk}\right]^{-1}\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n}+\alpha_{21}\Delta t k_1\right)\\
&\vdots\\
k_s &= -\left[\frac{dg_s\left(k^{(0)}\right)}{dk}\right]^{-1}\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n}+\alpha_{s,s-1}\Delta t k_{s-1}\right)\\
\mathbf{W}^{\mathrm{T}_{n+1}} &= \mathbf{W}^{\mathrm{T}_n}+\alpha_{s+1,s}\Delta t k_s.
\end{aligned}$$

Defining the updates $\mathbf{W}^{(0)} := \mathbf{W}^{\mathrm{T}_n}$ and

$$\mathbf{W}^{(j)} := \mathbf{W}^{\mathrm{T}_n}-\alpha_{j+1,j}\Delta t\left[\frac{dg_j\left(k^{(0)}\right)}{dk}\right]^{-1}\mathbf{M}^{-1}\mathbf{R}(\mathbf{W}^{(j-1)})$$

it can be shown by induction that the Runge-Kutta scheme given above can be reformulated equivalently by

$$\begin{aligned}
\mathbf{W}^{(0)} &:= \mathbf{W}^{\mathrm{T}_n}\\
\mathbf{W}^{(1)} &= \mathbf{W}^{(0)}-\alpha_{21}\Delta t\left[\frac{dg_1\left(k^{(0)}\right)}{dk}\right]^{-1}\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{(0)}\right)\\
&\vdots\\
\mathbf{W}^{(s)} &= \mathbf{W}^{(0)}-\alpha_{s+1,s}\Delta t\left[\frac{dg_s\left(k^{(0)}\right)}{dk}\right]^{-1}\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^{(s-1)}\right)\\
\mathbf{W}^{\mathrm{T}_{n+1}} &= \mathbf{W}^{(s)}.
\end{aligned} \tag{5.20}$$

Algorithm (5.20) indicates that for each stage the linear equation

$$\frac{dg_j\left(k^{(0)}\right)}{dk}\mathbf{h}_j = \alpha_{j+1,j}\Delta t\mathbf{M}^{-1}\mathbf{R}(\mathbf{W}^{(j-1)})$$

needs to be solved. This can be equivalently formulated by

$$\left((\Delta t)^{-1}\mathbf{M}+\alpha_{jj}\frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^{(j-1)}\right)\right)\mathbf{h}_j = \alpha_{j+1,j}\mathbf{R}(\mathbf{W}^{(j-1)}). \tag{5.21}$$

In a general context, Algorithm (5.20) may be interpreted as a kind of Rosenbrock method (see e.g. [24]). However, within this thesis we use this kind of method quite

differently, namely to approximate steady state solutions of (3.96a) and (3.96b), that is to approximately solve (3.97a) and (3.97b).

Therefore, we are not interested in time accurate methods. This allows us to modify the scheme using further acceleration techniques. First, for steady state computations the time step Δt in (5.21) is replaced (see also for example [41]) by some local time step $\Delta T := \operatorname{diag}(\operatorname{diag}(\Delta t_i)) \in \mathbb{R}^{N_{\mathrm{eq}}N \times N_{\mathrm{eq}}N}$. Here $N_{\mathrm{eq}} = 5$ for the mean flow equations and $N_{\mathrm{eq}} = N_t$ for the turbulence flow equations (see Section 2.2). As local time step we choose an approximation to the spectral radius of the diagonal blocks of $\frac{d\mathbf{R}}{d\mathbf{W}}$, more exact

$$\begin{aligned}\Delta t_i \;:=\; \mathrm{CFL}\cdot \mathrm{vol}\,(D_i) \Bigg[\sum_{j\in\mathcal{N}(i)} \mathrm{svol}(e_{ij}) \Bigg(\rho\left(\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_i}\right) \\ + C_v \rho \left(\frac{\partial \left\langle f_v\left(W_i, W_j\right), n_{e_{ij}} \right\rangle}{\partial W_i} \right)^{\mathrm{TSL},\mu=\mathrm{const}} \Bigg) \Bigg]^{-1}, \qquad C_v := 8.\end{aligned}$$

Using (4.13) together with Theorem 3.2.9 and Definition 3.2.3 as well as Theorem 4.3.4 we find the explicit expressions,

$$\begin{aligned}&\sum_{j\in\mathcal{N}(i)} \mathrm{svol}(e_{ij}) \left(\rho\left(\frac{\partial \mathcal{H}^{1st,\mathrm{Roe}}}{\partial W_i}\right) + \rho \left(\frac{\partial \left\langle f_v\left(W_i, W_j\right), n_{e_{ij}} \right\rangle}{\partial W_i} \right)^{\mathrm{TSL},\mu=\mathrm{const}} \right) \\ =\;& \sum_{j\in\mathcal{N}(i)} \mathrm{svol}(e_{ij}) \left(\lambda_{ij,\mathrm{Roe}} + \frac{C_v \mu_{\mathrm{eff},e_{ij}}}{\mathrm{dist}(e_{ij})\rho_i} \max\left\{ \frac{4}{3}, \frac{\kappa_{\mathrm{eff},e_{ij}}(\gamma-1)}{\mu_{\mathrm{eff},e_{ij}}} \right\} \right).\end{aligned}$$

As a further acceleration technique, to allow for over- and underrelaxation, we introduce a relaxation parameter ε, such that (5.21) is replaced by

$$\left((\Delta T)^{-1}\mathbf{M} + \varepsilon\alpha_{jj}\frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^{(j-1)}\right) \right) \mathbf{h}_j = \alpha_{j+1,j}\mathbf{R}(\mathbf{W}^{(j-1)}). \tag{5.22}$$

To shorten the notation, we close this section by defining the linear operator for stage j by

$$\mathbf{P}_j = (\Delta T)^{-1}\mathbf{M} + \varepsilon\alpha_{jj}\frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^{(j-1)}\right).$$

To formulate our final smoother for the nonlinear multigrid, we give a generalization of Newton's method using a so-called multistage implicit Runge-Kutta smoother, given by Algorithm 3.

Algorithm 3 Implicit Runge-Kutta smoother

1: $\mathbf{W}^{(0)} := \mathbf{W}^{\mathrm{T}_n}$
2: $\mathbf{W}^{(1)} = \mathbf{W}^{(0)} - \alpha_{21} \left(\mathbf{P}_1\right)^{-1} \mathbf{R}\left(\mathbf{W}^{(0)}\right)$
3: $\vdots$
4: $\mathbf{W}^{(s)} = \mathbf{W}^{(0)} - \alpha_{s+1,s} \left(\mathbf{P}_s\right)^{-1} \mathbf{R}\left(\mathbf{W}^{(s-1)}\right)$
5: $\mathbf{W}^{\mathrm{T}_{n+1}} = \mathbf{W}^{(s)}$

Theorem 5.1.8 *Assume that CFL* $\to \infty$, $s = 1$ *and* $\varepsilon = \alpha_{11} = \alpha_{21} = 1$. *Then Algorithm 3 is Newton's method.*

Proof: With respect to the assumptions formulated we obtain

$$\mathbf{W}^{\mathrm{T}_{n+1}} = \mathbf{W}^{\mathrm{T}_n} - \left(\frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^{\mathrm{T}_n}\right)\right)^{-1} \mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n}\right).$$

□

Theorem 5.1.8 sheds light on the fact that the derived Algorithm 3 represents only a generalization of Newton's method. Though we are in a position to formally apply Newton's method, let us shortly summarize the shortcomings of this method to motivate the necessity of Algorithm 3 in our context.

Assuming a good initial guess $\mathbf{W}^{\mathrm{T}_0}$ is given, the function $\mathbf{R}$ is smooth in a neighborhood of $\mathbf{W}^{\mathrm{T}_0}$, there exists a unique root of $\mathbf{R}$ and that we could easily solve the linear systems

$$\frac{d\mathbf{R}}{d\mathbf{W}}\mathbf{h}_n = \mathbf{R}\left(\mathbf{W}^{\mathrm{T}_n}\right), \qquad n = 0, 1, 2, \ldots, \tag{5.23}$$

then no research with respect to solution algorithms to solve the RANS equations (2.1a) would be required. On the other hand, none of the requirements stated above are satisfied in general, i.e.

a) The function $\mathbf{R}$ is not smooth.

b) The function $\mathbf{R}$ maybe has no root.

c) Even if the function $\mathbf{R}$ has a root, $\mathbf{W}^{\mathrm{T}_0}$ is not a good initial guess in general.

d) The linear systems (5.23) cannot be solved straightforward, it is even unknown if the matrices $\frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^n\right)$ are regular.

These are already enough reasons to note that a straightforward implementation of Newton's method within the context of the Reynolds averaged Navier-Stokes equations has nothing to do with a realistic assessment. And as a consequence, to implement a solution algorithm with no user interaction is much more a vision than a real world evaluation.

Moreover, it was not our intention to use Algorithm 3 as a solution method. In our context Algorithm 3 is applied as the smoother $\mathbf{S}$ in the Full Approximation Scheme multigrid described in Algorithm 5.1.6, which is also illustrated in Figure 5.1.

5.1.7 Globalization strategies

Though we are in a position to formally apply Newton's method (see Theorem 5.1.8), nothing has been said about the shortcomings of this method. One of the most severe shortcoming of Newton's method is that the linearization and convergence assertions and expectations are only true in a small neighborhood around the root. In practice the word "small" is useless, that is, in general it is impossible to decide during an iteration solving a nonlinear system of equation if an iterate is a neighborhood of the root. Hence, a strategy is required to find a good initial guess or vice versa a methodology is required to deal with an initial guess which is not in a small neighborhood of the root.

Strategy 1: Damping of correction, CFL strategy
Instead of trying to apply directly Newton's method we can replace the method by the algorithm

$$\left\| \frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^n\right)\mathbf{h}_n + \mathbf{R}\left(\mathbf{W}^n\right)\right\|^2 + \gamma_n \left\|\mathbf{h}_n\right\|^2 = \min_{h_n}! \tag{5.24a}$$

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \mathbf{h}_n. \tag{5.24b}$$

Here $\{\gamma_n\}_{n\in\mathbb{N}_0}$ denotes a sequence of parameters $\gamma_n \geq 0$. Algorithm 5.24 is called the Levenberg-Marquardt algorithm. Exploiting that the minimization problem (5.24a) is equivalent to

$$\left\| \begin{pmatrix} \frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^n\right) \\ \sqrt{\gamma_n}\mathbf{I} \end{pmatrix} \mathbf{h}_n + \begin{pmatrix} \mathbf{R}(\mathbf{W}^n) \\ 0 \end{pmatrix} \right\| = \min_{h_n}!,$$

the unique defined minimizer of (5.24a) can then be determined, for example, by solving the corresponding normal equation

$$\left(\gamma_n \mathbf{I} + \left(\frac{d\mathbf{R}}{d\mathbf{W}}\right)^T \frac{d\mathbf{R}}{d\mathbf{W}}\right)\mathbf{h}_n = -\left(\frac{d\mathbf{R}}{d\mathbf{W}}\right)^T \mathbf{R}\left(\mathbf{W}^n\right). \tag{5.25}$$

The Levenberg-Marquardt yields the inequality

$$\begin{aligned} \gamma_n \left\|\mathbf{h}_n\right\|^2 &\leq \left\| \frac{d\mathbf{R}}{d\mathbf{W}}\mathbf{h}_n + \mathbf{R}\left(\mathbf{W}^n\right)\right\|^2 + \gamma_n \left\|\mathbf{h}_n\right\|^2 \\ &= \min_{\mathbf{h}} \left[\left\| \frac{d\mathbf{R}}{d\mathbf{W}}\mathbf{h} + \mathbf{R}\left(\mathbf{W}^n\right)\right\|^2 + \gamma_n \left\|\mathbf{h}\right\|^2 \right] \\ &\leq \left\|\mathbf{R}\left(\mathbf{W}^n\right)\right\|^2, \end{aligned}$$

and therefore we get

$$\|\mathbf{h}_n\| \leq \frac{\|\mathbf{R}(\mathbf{W}^n)\|}{\sqrt{\gamma_n}}.$$

Hence, the parameter γ_n is a damping of the correction $\mathbf{h}_n$. It can be shown that the Levenberg-Marquardt algorithm converges for sufficiently large γ_n. On the other hand, if γ_n is large, the corrections are small and convergence may be arbitrarily slow. Comparing (5.25) and (5.21) gives rise to a new interpretation of the term $(\Delta T)^{-1}\mathbf{M}$. Though (5.21) neglects the additional multiplication by $\left(\frac{d\mathbf{R}}{d\mathbf{W}}\right)^T$, the operator $(\Delta T)^{-1}\mathbf{M}$ plays exactly the role of the stabilizing term $\gamma_n\mathbf{I}$, and the CFL-number is a weight for the corrections. This correspondence allows us to understand that in the initial phase of our solution scheme in general a small CFL number is required. Moreover, it also tells us that in general we cannot reach arbitrary large CFL numbers. For the problems considered it is during the iteration process in general impossible to say when the iterate is in a neighborhood of the solution, and hence some kind of damping is required.

For the Levenberg-Marquardt algorithm there exist methods to choose γ_n. For example, define

$$\varepsilon_n(\gamma_n) := \frac{\|\mathbf{R}(\mathbf{W}^n)\|^2 - \|\mathbf{R}(\mathbf{W}^n + \mathbf{h}_n)\|^2}{\|\mathbf{R}(\mathbf{W}^n)\|^2 - \left\|\mathbf{R}(\mathbf{W}^n) + \frac{d\mathbf{R}}{d\mathbf{W}}\mathbf{h}_n\right\|^2}.$$

One can say that $\varepsilon_n(\gamma_n)$ describes the change of the actual residual compared to its linearization. Hence, if $\varepsilon_n(\gamma_n)$ is small, it can be assumed that the trust region was too large, and the update $\mathbf{h}_n$ needs to be recomputed with a larger γ_n. Otherwise the step is accepted and γ_n might be reduced in the next step. Such a trust region consideration might be superior compared to a simple CFL number ramping,

$$\mathrm{CFL} = \min\left\{\mathrm{CFL}_{\mathrm{init}} \cdot f(n), \mathrm{CFL}_{\mathrm{max}}\right\}, \tag{5.26a}$$

$$f(n) = \begin{cases} 1, & n < 10, \\ \alpha^{n-10}, & n \geq 10, \end{cases} \qquad \alpha > 1. \tag{5.26b}$$

Moreover, the author is not aware of any publication where a Levenberg-Marquardt algorithm has been investigated for solving the discretized Reynolds-averaged Navier-Stokes equations. Naturally, when applying a Levenberg-Marquardt algorithm there is a significant increase in the computational complexity, since also the transposed derivatives are included into the algorithm. On the other hand the properties of this algorithm might be better suited than those of Newton's method.

Strategy 2: Runge-Kutta smoother

Using an implicit strategy to compute updates the actual choice of the Runge-Kutta smoother, that is the number of stages and the choice of the stage coefficients is an open problem. The multistage scheme can be interpreted as an admission on the two following facts:

a) The linear systems (5.21) cannot be solved up to a certain accuracy, or even if this is possible, the complexity to do this is in general so large, that the computational time required is unacceptable. And note, we are not interested in solving (5.21), but we are interested in solving the nonlinear problem (3.97).

b) Even if we had the exact solution of (5.21), we cannot expect that an undamped Newton method can be successful, in particular not in the start-up phase of the nonlinear iteration.

Hence, instead of a direct update the Runge-Kutta scheme is used as a further damping strategy, such that in the first stages the correction is weighted with a damping factor, and finally on the last stage the full correction is used for the update. It was shown in [44, 45] (see also Sections 7.2 and 7.3) and that this damping is even necessary in a neighborhood of the solution and allows for significant larger CFL numbers than using only a one-stage scheme. With a significant increase in the number of degrees of freedom it seems that such a damping is more important. Our conjecture is that it becomes more troublesome to solve within only a certain amount of complexity the linear systems (5.21) to get a reasonable update for the nonlinear iteration.

Strategy 3: Full multigrid
A further globalization strategy to find a suited initial guess is the application of full multigrid. Here, instead of directly starting the iterative nonlinear iteration on the finest grid level, subsequently starting on the coarsest grid level an approximate solution is computed and transferred to the next finer grid level. Hence, on the finest grid level the iterative process does not start from, for example, free stream values, but an interpolated approximate solution of the next coarser level. This is a well known procedure in the literature of multigrid and also well known in computational fluid dynamics. Though this is a powerful tool, to the author's experience this start up methodology is not used on a regular basis. If this observation is true, the reasons need to be found and clarification about the potential and possible shortcomings of this globalization strategy need to be investigated. For example, maybe for large scale applications the coarse grid quality is such that a stable discretization is not possible. Further research is required to improve full multigrid such that it can be used on regular basis for computational fluid dynamics computations.

Strategy 4: Treatment of turbulent flow equations
The treatment of the coupled system of equations (3.95) is an open problem. On the one hand, one can argue that (3.95) represents a coupled system of equations which needs to be solved all at once. On the other hand, the behavior of the turbulent flow equations during the iterative process is often significantly different compared to the mean flow equations. For example, the no-slip wall boundary condition for the ω equation of the $k\omega$-model (2.53) is not straightforward to implement. This is pointed out in Section 3.6. For this reason one has arguments to treat these

kinds of system of equations weakly coupled. There is not a satisfactory answer yet. From the global point of view, we want to mention that boundary conditions for the turbulent flow equations are often ad hoc. Furthermore, the initialization of the turbulent flow variables is to the author's point of view inappropriate. A remarkable reference for this observation is for example the numerical behavior of the normalized residual for the k-equation. The initial guess is so bad, such that the normalized residual increases after initialization several orders of magnitude before it starts to converge (see for example Figures 6.59 (left) and 6.63 (left)). So, the possibility to treat the turbulent flow equations in principle different than the mean flow equations might also be viewed as a globalization strategy. Section 6.11.2 presents in which way the weakly coupled algorithm can be exploited.

Strategy 5: There is no strategy

The discussed strategies 1–4 should not be misunderstood. All these are attempts to deal with the shortcomings of Newton's method. It can be assumed that this method can in general not be directly applied to approximate a solution of (3.97). The investigation and understanding of Newton's method and the methodologies to figure out ways to overcome its shortcomings is a long term topic of mathematics. It cannot be expected that for the solution of the Navier-Stokes and RANS equations, which are also not well understood from a mathematics point of view, there is an easy way out. The idea that there exists something like a "best practice", that is a classification of problems for which the same or a similar parameter choice work, must be handled with care. For example, it is well known for the RANS equations that small perturbations of the inflow conditions or small variations of the geometry may lead to totally different solutions. One may view such a design point as a critical point, for example the point of maximum lift. Often it is exactly such a design point which is of major interest. Unfortunately, exactly for these design points there is a good chance that the "best practice" strategy may fail. Mathematically spoken the problem is ill-posed, since there is no continuous dependency of the solution with respect to these inflow parameters and geometry variations. This conclusion makes it in particular important to improve numerical algorithms. On the other hand, in case the problem is well posed and there exists a continuous dependency of the solution with respect to the considered flow variations, then there is a good chance that a "best practice" strategy may work. Nevertheless, this discussion motivates that research and further investigations are required to develop a deep understanding of the components of solution algorithms and their interaction, to significantly improve the reliability of computational fluid dynamics software, such that it justifies industrial demands and can be used as a matter of course as a valuable tool in a process chain.

5.2 Linear solution methods

The linear equation (5.22) represents in general a large scale, ill-conditioned system and cannot be solved directly. Matrix-free Krylov subspace methods are therefore a natural choice to approximate a solution of (5.22) within a small number of steps.

5.2.1 Krylov subspace methods

At the outset there exist several Krylov subspace methods. Generally spoken, all these methods are based on the same idea. They construct a certain Krylov subspace. In this subspace an approximate solution to the given linear system is computed. The quality of this approximation depends on this subspace and the right hand side of the linear system.

It is the author's opinion that in general not the particular choice of a Krylov subspace method is important (see for example the textbook of Saad [84] for several such methods), but that the construction of a well suited preconditioner is of major significance. Thus, instead of considering a variety of Krylov subspace methods we restrict ourselves to the Generalized Minimum Residual (GMRES) method preconditioned from the left, that is we apply this method to obtain

$$\mathbf{Prec}_j^{-1}\mathbf{P}_j\mathbf{x} = \alpha_{j+1,j}\mathbf{Prec}_j^{-1}\mathbf{R}(\mathbf{W}^{(j-1)}). \tag{5.27}$$

The GMRES algorithm applied to (5.27) with initial guess $\mathbf{x}^{(0)}$ reads as follows.

Algorithm 5.2.1 *Left preconditioned GMRES method applied to (5.27):*

- *Solve (approximately)* $\mathbf{Prec}_j\mathbf{r}_0 = \alpha_{j+1,j}\mathbf{R}\left(\mathbf{W}^{(j-1)}\right) - \mathbf{P}_j\mathbf{x}^{(0)}$
- *Compute* $\beta := \|\mathbf{r}_0\|_2, \mathbf{v}_1 := \frac{1}{\beta}\mathbf{r}_0$
- *for* $k = 1, \ldots, m$
  - *Solve (approximately)* $\mathbf{Prec}_j\mathbf{w} = \mathbf{P}_j\mathbf{v}_k$
  - *for* $i = 1, \ldots k$
    - $h_{i,k} := \langle \mathbf{w}, \mathbf{v}_i \rangle$
    - $\mathbf{w} := \mathbf{w} - h_{i,k}\mathbf{v}_i$
  - $h_{k+1,k} = \|\mathbf{w}\|_2, \mathbf{v}_{k+1} = \frac{1}{h_{k+1,k}}\mathbf{w}$
- $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_m), H_m = (h_{i,k})_{1 \le i \le k+1, 1 \le k \le m}$
- *Solve* $\mathbf{y}_m := \mathrm{argmin}_{\mathbf{y}}\|\beta e_1 - H_m\mathbf{y}\|_2$ *by Given's-rotations*
- $\mathbf{x}^{(m)} := \mathbf{x}^{(0)} + \mathbf{V}_m\mathbf{y}_m$

A preconditioner $\mathbf{Prec}_j$ should satisfy the following conditions:

- $\mathbf{Prec}_j \approx \mathbf{P}_j$, that is $\mathbf{Prec}_j$ should be a good approximation to $\mathbf{P}_j$.
- The storage requirement for $\mathbf{Prec}_j$ should be acceptable.
- The system $\mathbf{Prec}_j\mathbf{w} = \mathbf{r}$ must be efficiently (approximately) solvable.

For the implementation of GMRES it is worthwhile to mention that the modified Gram-Schmidt orthogonalization within the Arnoldi process is often not stable, and it cannot be assumed to be stable for the problems at hand. This is in particular true for ill-conditioned matrices [71, 40]. So, instead a GMRES with orthogonalization based on Householder reflections might be a better choice [84]. Moreover, also the reduction of the upper Hessenberg matrix H_m is done on-the-fly during the algorithm as reported in [84].

5.2.2 Construction of preconditioner

From the equation for $\mathbf{w}$ of the GMRES Algorithm 5.2.1 above it is obvious that we need to approximately solve the linear systems

$$\mathbf{Prec}_j\mathbf{w} = \mathbf{P}_j\mathbf{v}_k. \tag{5.28}$$

So, the preconditioner we are going to design consists of two parts:

a) the linear operator $\mathbf{Prec}_j$ itself,

b) an iterative solution method for approximately solving the linear systems (5.28) required for GMRES.

Instead of taking the exact residual $\mathbf{R}$ for the Jacobian a simplification satisfying $\tilde{\mathbf{R}} \approx \mathbf{R}$ is considered to construct the preconditioner for (5.27). Here we consider a compact stencil (see Notation 4.1.5) approximating the extended stencil, that is we choose $\tilde{\mathbf{R}} = \mathbf{R}^{\text{comp}} \approx \mathbf{R}$. Considering compact discretization has for the derivative the major advantage that the stencil at point i relies only on next neighbor information. The associated Jacobian $\frac{d\mathbf{R}^{\text{comp}}}{d\mathbf{W}}$ has several properties of interest (see Section 4.1), for example:

a) It is much less memory intensive than the exact derivative $\frac{d\mathbf{R}}{d\mathbf{W}}$.

b) It can be constructed by a loop over all edges corresponding to the design of the residual evaluation $\mathbf{R}^{\text{comp}}$.

c) Assuming that $\mathbf{R} \approx \mathbf{R}^{\text{comp}}$ holds, it can be assumed that $\frac{d\mathbf{R}}{d\mathbf{W}} \approx \frac{d\mathbf{R}^{\text{comp}}}{d\mathbf{W}}$.

To construct a preconditioner $\mathbf{Prec}_j$ for the mean flow equations $\mathbf{R}_{\text{mean}}$ we use (4.13) as approximation for the inviscid and (4.34) and (4.35) for the viscous part. A corresponding discretization is constructed by the assumptions that $\left|\mathbf{A}_{ij}^{\text{Roe}}\right| = const$, $\mu_{\text{eff},e_{ij}} = const.$ and gradients are computed using a TSL approximation. We express such a residual by

$$\tilde{\mathbf{R}}_{\text{prec}}^{\text{comp}}(\mathbf{W}) = \tilde{\mathbf{R}}_{\left|\mathbf{A}_{ij}^{\text{Roe}}\right|=const.,\mu_{\text{eff},e_{ij}}=const.}^{\text{comp}}(\mathbf{W}).$$

Using this notation and the corresponding approximate derivatives we need to include the stabilizing terms such that the final preconditioner is given by

$$\mathbf{Prec}_j := (\Delta T)^{-1}\mathbf{M} + \varepsilon\alpha_{jj}\frac{d\tilde{\mathbf{R}}_{\text{prec}}^{\text{comp}}}{d\mathbf{W}}\left(\mathbf{W}^{(j-1)}\right). \tag{5.29}$$

The preconditioner $\mathbf{Prec}_{j,\text{SA}}$ for the Spalart-Allmaras turbulence model uses the derivatives for inviscid, viscous and source terms in the way they are presented in Section 4.4.1. Hence, in case one uses TSL approximations for the gradients the $\mathbf{Prec}_{j,\text{SA}}$ is the exact derivative of the considered discretization, denoted by

$$\mathbf{Prec}_{j,\text{SA}} = (\Delta T)^{-1}\mathbf{M} + \varepsilon\alpha_{jj}\frac{d\tilde{\mathbf{R}}_{\text{prec}}^{\text{comp,SA}}}{d\mathbf{W}}\left(\tilde{\nu}^{(j-1)}\right). \tag{5.30}$$

The preconditioner $\mathbf{Prec}_{j,(k,\omega)}$ for the $k\omega$-turbulence model uses for the inviscid and viscous part the derivatives stated in Section 4.4.2. The derivatives of the source terms need to be modified. The diagonal terms of the derivatives of the destruction terms are neglected, that is

$$\frac{\partial De_{k,(k,\omega)}}{\partial k_i} \qquad \text{and} \qquad \frac{\partial De_{\omega,(k,\omega)}}{\partial \omega_i}$$

are left out in the preconditioner. The necessity for this modification is discussed in Section 7.4. Corresponding to (5.29) and (5.30) the preconditioner for the $k\omega$-model is given by

$$\mathbf{Prec}_{j,(k,\omega)} = (\Delta T)^{-1}\mathbf{M} + \varepsilon\alpha_{jj}\frac{d\tilde{\mathbf{R}}_{\text{prec}}^{\text{comp},(k,\omega)}}{d\mathbf{W}}\left((k,\omega)^{(j-1)}\right). \tag{5.31}$$

For completeness, we present a further preconditioner well known in the literature of computational fluid dynamics [116]. It is based on a further simplification of the Jacobian and the major idea is to replace the operational count. This is realized by approximating the local block systems by their spectral radius. Hence, instead of (4.13) we approximate

$$\sum_{j\in\mathcal{N}(i)}\frac{\partial\mathcal{H}^{1st,\text{Roe}}}{\partial W_k} \approx \frac{1}{2}\begin{cases} \sum_{j\in\mathcal{N}(i)}\rho\left(\left|\mathbf{A}_{ij}^{\text{Roe}}\right|\right), & k=i, \\ -\rho\left(\left|\mathbf{A}_{ij}^{\text{Roe}}\right|\right), & k\in\mathcal{N}(i), \\ 0, & k\neq i, k\notin\mathcal{N}(i). \end{cases} \tag{5.32}$$

Such an approximate derivative corresponds for example to a scalar dissipative or van Leer flux. The spectral radius is given by $\lambda_{ij,\mathrm{Roe}}$ denoted in Definition 3.2.3. Furthermore, the viscous derivatives are also approximated by their spectral radii, that is (4.34) and (4.35) are replaced by

$$\frac{\partial \mathbf{R}_i^{\mathrm{visc,scalar}}}{\partial \mathbf{W}_k} \approx \begin{cases} \sum_{j\in\mathcal{N}(i)} \rho\left(\left(\frac{\partial\langle f_v(W_i,W_j),n\rangle}{\partial W_i}\right)^{\mathrm{TSL},\mu=\mathrm{const}}\right), & k=i, \\ -\rho\left(\left(\frac{\partial\langle f_v(W_i,W_j),n\rangle}{\partial W_i}\right)^{\mathrm{TSL},\mu=\mathrm{const}}\right), & k\in\mathcal{N}(i), \\ 0, & \text{else.} \end{cases} \tag{5.33}$$

The spectral radii can be computed using Theorem 4.3.4 (see also [47]). Obviously, such a preconditioner does not require memory to save a whole block CSR matrix (see Section 4.1.1), but only scalar values. This idea can also be carried over to the turbulence flow equations. In particular, for these equations we only take the diagonal part of the derivative of the source terms. The simplifications yield a much more memory efficient method. On the other hand, estimating the Jacobian by its spectral radius can have rather serious detrimental effects on the convergence of an implicit scheme, for example much slower convergence rates. In particular, for mesh refinement studies and more complex flow cases this statement is confirmed in [48]. For later use we denote the preconditioner corresponding to these simplifications by $\mathbf{Prec}_j^{\mathrm{scalar}}$. To justify that $\mathbf{Prec}_j^{\mathrm{scalar}}$ and $\mathbf{Prec}_j$ may be used as preconditioners we assume that the following approximation properties hold:

$$\mathbf{Prec}_j^{\mathrm{scalar}} \approx \mathbf{Prec}_j \approx \mathbf{P}_j.$$

5.2.3 Iterative solution methods for linear equations

Another ingredient to include with a powerful preconditioner is to approximate efficiently a solution of the linear systems (5.28). The notation $\mathbf{Prec}_j$ in (5.28) is now understood as surrogate for one of the possible linear operators designed in Section 5.2.2. To approximately solve efficiently the linear system (5.28) linear multigrid methods can be applied. Illustrated in Figure 5.1, we need to define the most inner two boxes.

To formulate a linear multigrid algorithm we consider the notation used in Section 5.1.3 and define corresponding to (5.2)

$$A_{M_k} := \mathbf{Prec}_j : \mathbb{R}^{\#M_k} \to \mathbb{R}^{\#M_k}, \qquad A_{M_k} x_{M_k} = b_{M_k}.$$

Algorithm 5.2.2 *Assume that a sequence of meshes $M_n \subset \ldots \subset M_1$ and the linear equations*

$$A_{M_k} x_{M_k}^{(\ell)} = b_{M_k}$$

and an approximate solution $x_{M_k}^{(\ell)}$ *together with a smoother* $\mathbf{S} = \mathbf{S}\left(x_{M_k}^{(\ell)}, A_{M_k}, b_{M_k}\right)$ *are given. Here* $\mathbf{S}$ *denotes a method which determines from the approximate solution* $x_{M_k}^{(\ell)}$ *an update*

$$\overline{x}_{M_k}^{(\ell)} := \mathbf{S}\left(x_{M_k}^{(\ell)}, A_{M_k}, b_{M_k}\right).$$

Let us denote by

- $\omega_{M_k,\mathrm{pre}} \in \mathbb{N}_0$ *the number of presmoothing steps*
- $\omega_{M_{k+1}} \in \mathbb{N}_0$ *the number of coarse grid smoothing steps*
- $\omega_{M_k,\mathrm{post}} \in \mathbb{N}_0$ *the number of postsmoothing steps*

Then we define a $2V = 2V\left(k, \omega_{M_k,\mathrm{pre}}, \omega_{M_{k+1}}, \omega_{M_k,\mathrm{post}}\right)$ *multigrid cycle by*

a) Apply $\omega_{M_k,\mathrm{pre}}$ *smoothing steps, that is compute*

$$\overline{x}_{M_k}^{(\ell)} = \mathbf{S} \circ \cdots \circ \mathbf{S}\left(x_{M_k}^{(\ell)}, A_{M_k}, b_{M_k}\right) = \mathbf{S}^{\omega_{M_k,\mathrm{pre}}}\left(x_{M_k}^{(\ell)}, A_{M_k}, b_{M_k}\right).$$

b) Compute the right hand side of the coarse grid equation

$$b_{M_{k+1}} = P_{M_{k+1}}^{M_k}\left(b_{M_k} - A_{M_k}\left(\overline{x}_{M_k}^{(\ell)}\right)\right).$$

c) Apply $\omega_{M_{k+1}}$ *coarse grid smoothing steps, that is compute*

$$\overline{x}_{M_{k+1}}^{(\ell)} = \mathbf{S}^{\omega_{M_{k+1}}}\left(P_{M_{k+1}}^{M_k}\overline{x}_{M_k}^{(\ell)}, A_{M_{k+1}}, b_{M_{k+1}}\right).$$

d) Compute an update of the approximate solution

$$\overline{x}_{M_k}^{(\ell+1)} := \overline{x}_{M_k}^{(\ell)} + I_{M_{k+1}}^{M_k}\overline{x}_{M_{k+1}}^{(\ell)}.$$

e) Apply $\omega_{M_k,\mathrm{post}}$ *smoothing steps, that is compute*

$$x_{M_k}^{(\ell+1)} = \mathbf{S}^{\omega_{M_k,\mathrm{post}}}\left(\overline{x}_{M_k}^{(\ell+1)}, A_{M_k}, b_{M_k}\right).$$

The linear multigrid Algorithm 5.2.2 can be derived by an application of the Full Approximation Scheme Algorithm 5.1.6 to a linear equation in the following way. Assuming that $\mathbf{R}_{M_k}$ is a linear operator, we rewrite the coarse grid equation by

$$\mathbf{R}_{M_{k+1}}\left(\mathbf{U}_{M_{k+1}}^{(\ell)} - P_{M_{k+1}}^{M_k}\overline{\mathbf{U}}_{M_k}^{(\ell)}\right) = P_{M_{k+1}}^{M_k}\left(\mathbf{f}_{M_k} - \mathbf{R}_{M_k}\overline{\mathbf{U}}_{M_k}^{(\ell)}\right).$$

Hence, substituting $\mathbf{V}_{M_{k+1}}^{(\ell)} = \mathbf{U}_{M_{k+1}}^{(\ell)} - P_{M_{k+1}}^{M_k} \overline{\mathbf{U}}_{M_k}^{(\ell)}$ we get after application of $\omega_{M_{k+1}}$ smoothing steps

$$\overline{\mathbf{V}}_{M_{k+1}}^{(\ell)} = \mathbf{S}^{\omega_{M_{k+1}}} \left(\mathbf{V}_{M_{k+1}}^{(\ell)}, \mathbf{R}_{M_{k+1}}, \mathbf{f}_{M_{k+1}} \right),$$

which yields the correction

$$\overline{\mathbf{V}}_{M_{k+1}}^{(\ell)} = \overline{\mathbf{U}}_{M_{k+1}}^{(\ell)} - P_{M_{k+1}}^{M_k} \overline{\mathbf{U}}_{M_k}^{(\ell)}.$$

and the update for the fine grid iterate

$$\overline{\mathbf{U}}_{M_k}^{(\ell+1)} = \overline{\mathbf{U}}_{M_{k+1}}^{(\ell)} + I_{M_{k+1}}^{M_k} \overline{\mathbf{V}}_{M_{k+1}}^{(\ell)}.$$

These considerations give exactly the linear multigrid method Algorithm 5.2.2, and it shows that for a linear operator Algorithm 5.1.6 is equivalent to Algorithm 5.2.2.

We now turn to the task to design a smoother for the linear multigrid method. Here we consider straightforward iterative solution methods, for example:

- (Block) Jacobi,
- (Block) Gauss-Seidel,
- symmetric (Block) Gauss-Seidel.

However, $\mathbf{Prec}_j$ is in general neither symmetric nor it can be assumed to be block diagonal dominant. So, in particular for high Reynolds number viscous flows, where meshes with large anisotropies in the boundary layer are required, problem adapted iterative methods are necessary. Exploiting line information acting in the direction of strong coupling, these iterative solution methods may be significantly improved. To construct such information was topic of Section 5.1.1.

Using the Notation 5.1.2, along the line L_i the corresponding (block) tridiagonal matrix $\mathbf{Tri}_{L_i} \in \mathbb{R}^{(N_{\mathrm{eq}} r_i) \times (N_{\mathrm{eq}} r_i)}$ of $A_{M_k} = (A_{ij})$, $A_{ij} \in \mathbb{R}^{N_{\mathrm{eq}} \times N_{\mathrm{eq}}}$ is given by

$$\mathbf{Tri}_{L_i} := \begin{pmatrix} A_{\ell_i^1,\ell_i^1} & A_{\ell_i^1,\ell_i^2} & & & & \\ A_{\ell_i^2,\ell_i^1} & A_{\ell_i^2,\ell_i^2} & A_{\ell_i^2,\ell_i^3} & & & \\ & \ddots & \ddots & \ddots & & \\ & & A_{\ell_i^{r_i-1},\ell_i^{r_i-2}} & A_{\ell_i^{r_i-1},\ell_i^{r_i-1}} & A_{\ell_i^{r_i-1},\ell_i^{r_i}} & \\ & & & A_{\ell_i^{r_i},\ell_i^{r_i-1}} & A_{\ell_i^{r_i},\ell_i^{r_i}} & \end{pmatrix}. \tag{5.34}$$

In case $r_i = 1$ the matrix $\mathbf{Tri}_{L_i}$ simplifies to a diagonal block. Exploiting (5.34) possible iterative solution methods to approximate a solution of

$$A_{M_k} x = b$$

are relaxed Block line Jacobi (5.35) and Block line Gauss-Seidel (5.36) method:

$$x_{L_i}^{(k+1)} = (1-\omega_r)\, x_i^{(k)} + \omega_r \mathbf{Tri}_{L_i}^{-1} \left(b_{L_i} - \sum_{j \in L_1,\ldots,L_n, j \notin G_i} A_{L_i,j} x_j^{(k)} \right), \quad (5.35)$$

$$x_{L_i}^{(k+1)} = (1-\omega_r)\, x_{L_i}^{(k)} + \omega_r \mathbf{Tri}_{L_i}^{-1} \left(b_{L_i} - \sum_{j \in L_1,\ldots,L_{i-1}, j \notin L_i} A_{L_i,j} x_j^{(k+1)} - \sum_{j \in \{1,\ldots N\} \setminus \{L_1,\ldots,L_i\}} A_{L_i,j} x_j^{(k)} \right), \quad (5.36)$$

where we iterate over all Lines L_i, $i = 1, \ldots, n$. In case line information is not available these methods reduce to the well known relaxed Block Jacobi (5.37) and Block Gauss-Seidel (5.38) method, pointwise written for $i = 1, \ldots, N_{elem}$:

$$x_i^{(k+1)} = (1-\omega_r)\, x_i^{(k)} + \omega_r \left(A_{i,i}\right)^{-1} \left(b_i - \sum_{j=1, j \neq i}^{N_{elem}} A_{i,j} x_j^{(k)} \right), \quad (5.37)$$

$$x_i^{(k+1)} = (1-\omega_r)\, x_i^{(k)} + \omega_r \left(A_{i,i}\right)^{-1} \left(b_i - \sum_{j=1}^{i-1} A_{i,j} x_j^{(k+1)} - \sum_{j=i+1}^{N_{elem}} A_{i,j} x_j^{(k)} \right) \quad (5.38)$$

Definition 5.2.3 *In case we apply for (5.36) and (5.38) a forward sweep followed by a backward sweep, we call these methods Block symmetric line Gauss-Seidel and Block symmetric Gauss-Seidel, respectively.*

Remark 5.2.4 *Let us comment on some implementation details for (5.35) – (5.38):*

a) *Naturally, within the multigrid context we do not consider these methods as iterative solvers but as smoothers* $\mathbf{S}$ *applied in Algorithm 5.2.2 using an early stopping criterion.*

b) *For all methods we choose as initial guess* $x^{(0)} = 0$.

c) *Only (5.35) and (5.36) need to be implemented, (5.37) and (5.38) follow by specialization, that is neglecting line information.*

d) *The tridiagonal blocks* $\mathbf{Tri}_{L_i}$ *and diagonal blocks* $A_{i,i}$ *are not inverted, but only the block LU-decomposition of* $\mathbf{Tri}_{L_i}$ *and the LU-decomposition of* $A_{i,i}$ *are computed exploiting pivot techniques. Then the corresponding linear systems are solved.*

e) *A symmetric line sweep is done as follows: First we apply the sweep over all lines. In a parallel environment we then exchange the approximate solution*

between the partitions. Then we proceed with the sweep over all remaining points, which is again followed by communication of the approximate solution. To close the symmetric sweep we do everything in the opposite manner. At the beginning, we sweep over all points which do not correspond to a line in the reverse manner followed by communication of the approximate solution, then the lines follow in opposite ordering closed by communicating the approximate solution. Actually, the algorithm is line Gauss-Seidel only at the level between partitions. We try to overcome this deficiency by communicating between the domains several times.

f) Implementation of multi-coloring algorithms has not been considered so far and is part of future work.

g) The coarsening strategy described in Section 5.1.2 ensures that lines required to set up the tridiagonal systems (5.34) are retained on each multigrid level. Of course, if on some multigrid level a line degenerates into a point, this line is cleared from the set of lines and treated as a usual point in the following coarsening steps. With this strategy, arbitrary coarse multigrid levels can be constructed. In a parallel environment multigrid levels are formed within each partition.

Finally, (5.35) – (5.38) need to be combined with a suitable stopping criterion. Here we suggest:

$$f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, \max_{\text{iter}}, \varepsilon(k)\right) = \begin{cases} \text{true}, & \frac{\|b - A_{M_k} x^{(k)}\|_2}{\|b\|_2} < \varepsilon(k), \\ \text{true}, & k \geq \max_{\text{iter}}, \\ \text{false}, & \text{else}. \end{cases} \tag{5.39}$$

In case we want to stop with a fixed number of iterations we choose $\varepsilon(k) = -1$ in (5.39).

5.3 Hierarchy of multigrid smoothers

Algorithms approximating solutions of the Reynolds averaged Navier-Stokes equations (2.1a) and (2.45) should follow the goal to reduce the user interaction with the software to a minimum. On the other hand the literature about computational fluid dynamics is full of suggestions of different solution algorithms and the authors often claim that the new developed algorithm is superior with respect to some certain property. Here, it is not our goal to develop a new solution algorithm, which is in our context a smoother for the nonlinear multigrid. But it is our goal to show that almost all suggestions found in the literature are variants of the algorithms suggested so far.

5.3.1 Low cost smoothers

To obtain several low cost solution methodologies suggested in the literature about computational fluid dynamics we now suggest the following simplification: The number of steps in the Krylov subspace method is reduced to zero. Then, by a view on Algorithm 5.2.1, only the first preconditioning step is left, i.e. Algorithm 5.2.1 reduces to approximate the solution of

$$\mathbf{Prec}_j \mathbf{h}_j = \alpha_{j+1,j} \mathbf{R}\left(\mathbf{W}^{(j-1)}\right), \tag{5.40}$$

and as a consequence Algorithm 3 simplifies to

$$\begin{aligned} \mathbf{W}^{(0)} &:= \mathbf{W}^n \\ \mathbf{W}^{(1)} &= \mathbf{W}^{(0)} - \alpha_{21}\mathbf{Prec}_1^{-1}\mathbf{R}\left(\mathbf{W}^{(0)}\right) \\ &\vdots \\ \mathbf{W}^{(s)} &= \mathbf{W}^{(0)} - \alpha_{s+1,s}\mathbf{Prec}_s^{-1}\mathbf{R}\left(\mathbf{W}^{(s-1)}\right) \\ \mathbf{W}^{n+1} &= \mathbf{W}^{(s)}. \end{aligned} \tag{5.41}$$

The Runge-Kutta iteration (5.41) depends on the construction of **Prec** and the iterative linear solution method, e.g. (5.35) – (5.38). A further, self-evident alternative of algorithm (5.41) is the freezing of the preconditioner **Prec** on the first stage, that is

$$\begin{aligned} \mathbf{W}^{(0)} &:= \mathbf{W}^n \\ \mathbf{W}^{(1)} &= \mathbf{W}^{(0)} - \alpha_{21}\mathbf{Prec}_1^{-1}\mathbf{R}\left(\mathbf{W}^{(0)}\right) \\ &\vdots \\ \mathbf{W}^{(s)} &= \mathbf{W}^{(0)} - \alpha_{s+1,s}\mathbf{Prec}_1^{-1}\mathbf{R}\left(\mathbf{W}^{(s-1)}\right) \\ \mathbf{W}^{n+1} &= \mathbf{W}^{(s)}. \end{aligned} \tag{5.42}$$

Assuming that the operator $\mathbf{Prec}_j$, $j = 1, \ldots, s$, do not change significantly over one Runge-Kutta iteration and that the construction of $\mathbf{Prec}_j$ together with the computation of the block LU-decomposition of $\mathbf{Tri}_{L_i}$ is a time consuming approach, this simple frozen Runge-Kutta iteration may yield an efficient alternative compared with algorithm (5.41). Example 1 in Section 6.3 is dedicated a corresponding investigation.
Several variants of the general formulations (5.41) and (5.42) correspond to well known suggested solution techniques given in the literature.

Theorem 5.3.1 *Consider Algorithm (5.41) or (5.42) as smoother* **S** *for Algorithm 5.1.6 and* $\mathbf{Prec}_j$ *is constructed as described in Section 5.2.2. Approximately solving the linear systems (5.40) by application of Algorithm 5.2.2 with* $n = 1$ *together with a symmetric (line) Gauss-Seidel method and choosing*

$$f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, N, -1\right), \qquad N > 1,$$

we get exactly a solution algorithm denoted as RK/Implicit smoother or first order preconditioned Runge-Kutta method suggested for example in [80, 93, 44].

Proof: Choosing $n = 1$ reduces Algorithm 5.2.2 to a single grid iteration. The choice of $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, N, -1\right)$, means that the iterative solver is truncated after a fixed number of iterations, e.g. $N = 2, \ldots, 5$ is suggested in the mentioned literature. Then the theorem follows by a straightforward comparison of the methods.

□

Theorem 5.3.2 *Consider Algorithm (5.41) or (5.42) as smoother* $\mathbf{S}$ *for Algorithm 5.1.6 and* $\mathbf{Prec}_j$ *is constructed as described in Section 5.2.2. Approximately solving the linear systems (5.40) by application of Algorithm 5.2.2 with* $n = 1$ *together with a line Jacobi method and choosing* $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, 1, -1\right)$*, we get exactly a solution algorithm denoted as line implicit method suggested for example in [74, 57, 16, 43].*

Proof: Choosing $n = 1$ reduces Algorithm 5.2.2 to a single grid iteration. The choice of $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, 1, -1\right)$, means that the line Jacobi method (5.35) is stopped after the first iteration. Considering as initial guess $x^{(0)} = 0$, the approximate solution is obtained by

$$x_{L_i}^{(1)} = \mathbf{Tri}_{L_i}^{-1} b_{L_i}.$$

Then the theorem follows by a straightforward comparison of the methods.

□

Theorem 5.3.3 *Consider Algorithm (5.41) or (5.42) as smoother* $\mathbf{S}$ *for Algorithm 5.1.6 and* $\mathbf{Prec}_j$ *is constructed as described in Section 5.2.2. Approximately solving the linear systems (5.40) by application of Algorithm 5.2.2 with* $n = 1$ *together with a Jacobi method and choosing* $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, 1, -1\right)$*, we get exactly a solution algorithm denoted as point implicit method suggested for example in [73, 63, 42].*

Proof: Choosing $n = 1$ reduces Algorithm 5.2.2 to a single grid iteration. The choice of $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, 1, -1\right)$, means that the Jacobi method (5.35) is stopped after the first iteration. Considering as initial guess $x^{(0)} = 0$, the approximate solution is obtained by

$$x_i^{(1)} = \left(A_{i,i}\right)^{-1} b_i.$$

Then the theorem follows by a straightforward comparison of the methods.

□

Theorem 5.3.4 *Consider Algorithm (5.41) or (5.42) as smoother* $\mathbf{S}$ *for Algorithm 5.1.6 and* $\mathbf{Prec}_j^{\text{scalar}}$ *is constructed as described in Section 5.2.2. Approximately solving the linear systems (5.40) by application of Algorithm 5.2.2 with* $n = 1$ *together with a symmetric Gauss-Seidel method and choosing the stopping criterion by* $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, 1, -1\right)$*, we get exactly a solution algorithm denoted as LU-SGS method suggested for example in [116].*

Proof: Choosing $n = 1$ reduces Algorithm 5.2.2 to a single grid iteration. The choice of $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, 1, -1\right)$, means that the symmetric Gauss-Seidel method is stopped after the first iteration, that is one forward and one backward sweep. Then the theorem follows by a straightforward comparison of the methods.

□

Theorem 5.3.5 *Consider Algorithm (5.41) or (5.42) as smoother* $\mathbf{S}$ *for Algorithm 5.1.6 and* $\mathbf{Prec}_j$ *or* $\mathbf{Prec}_j^{\text{scalar}}$ *is constructed as described in Section 5.2.2 with* $\alpha_{jj} = 0$*. Approximately solving the linear systems (5.40) by application of Algorithm 5.2.2 with* $n = 1$ *together with a Jacobi method and choosing the stopping criterion by* $f_{\text{stop}}\left(A_{M_k}, x^{(k)}, b, k, 1, -1\right)$ *we get exactly a solution algorithm denoted as explicit Runge-Kutta accelerated by local time stepping [31].*

Proof: Choosing $n = 1$ reduces Algorithm 5.2.2 to a single grid iteration. Exploiting that $\alpha_{jj} = 0$ and one Jacobi iteration with initial guess $x^{(0)} = 0$, the approximate solution of equation (5.40) is given by

$$\mathbf{h}_j = \Delta T \mathbf{M}^{-1} \mathbf{R}\left(\mathbf{W}^{(j-1)}\right).$$

Then the theorem follows by a straightforward comparison of the methods.

□

This direct comparison of the different methods gives rise to their capability to solve (3.97) for high Reynolds number viscous flows on anisotropic meshes. And, in general there is no justification for the simplifications done in the methods above. Hence, though these methods are proposed throughout the literature of computational fluid dynamics, with respect to the significant increase in mesh sizes and complexity of flow physics we predict the methods mentioned in Theorems 5.3.2 – 5.3.5 only limited potential. In [45] and Sections 7.2 and 7.3 analytical reasons are given. Examples considered in Chapter 6 give further indication for this statement.

5.3.2 The solution algorithm: A castle in a sand pit

Having discussed the possible variants of possible smoothers for the nonlinear multigrid, let us come back to the global view given in Figure 5.1. We summarize several variants which can be chosen.

1) **Nonlinear multigrid**
   a) Agglomeration
      * Overall coarsening ratio
      * Semi / directional coarsening strategy
      * Advancing front or weighted graph algorithm
   b) Projection
   c) Interpolation
2) **Runge-Kutta smoother**
   a) Number of stages
   b) Stage coefficients
   c) CFL number
3) **Krylov subspace method**
   a) Full Orthogonalization Method, Generalized Minimum Residual Method, Biconjugate Gradient Stabilized Algorithm, Conjugate gradient for the normal equations, ...
   b) Number of iterations, residual based truncation criterion
4) **Linear Preconditioner**
   a) Construction of preconditioning matrix
      * Approximate Jacobian
      * Deflation based preconditioner
      * Polynomial preconditioner
   b) Truncation criterion for approximate linear solve
5) **Linear Multigrid**
   a) Agglomeration multigrid
      * Overall coarsening ratio
      * Semi / directional coarsening strategy
      * Advancing front or weighted graph algorithm
   b) Algebraic multigrid
   c) Smoother
      * Gauss-Seidel type smoother

    - ILU decomposition

d) Projection

e) Interpolation

It is out of the scope of this thesis to investigate and compare all the variants of algorithms which can be set up by a combination of all these different algorithms and parameters. Moreover, note that all these subitems in general lead to a further parameter choice as well as subvariants.

Coming back to the topic about requirements for a modern software package solving the RANS equations (2.1a) or (2.45) it is impossible to give general recommendations. Here, we have shown that certain parameter choices yield methods of totally different character. We can choose between Newton's method, which is represented in general by some kind of regularized Newton method, and explicit Runge-Kutta methods with local time stepping.

Since there is no clear recommendation at this point in time, a modern computer code should be implemented such that all these variants of algorithms need to be easily constructed. That is the code design should be in such a way that a powerful linear algebra package represents the main infrastructure of the code. This infrastructure is then used to combine the different matrix vector operations, also including the variants of linear solution algorithms, such that a flexible application of all these possibilities is feasible. From both a practical point of view and scientific point of view it is future work to reduce the number of variants and parameters. Otherwise it is impossible to offer a software reducing the user interaction to a minimum.

It is the strong belief of the author of this thesis, that with an increase of the complexity of the simulations going hand in hand with a significant increase of the mesh size and degrees of freedom, simplifications of the smoothers in the sense that they are not close to some regularized Newton method are not suitable to find approximate solutions of the RANS equations.

Chapter 6

Examples

Within this thesis we have touched two major problems of computational fluid dynamics. The first, and major issue of this thesis is the development and investigation of a suitable solution method for the RANS equations and corresponding boundary value problems. In the Introduction we mentioned that in our understanding this wording comprises both the discretization strategy and a corresponding solution method for the resulting set of algebraic equations. Before demonstrating several examples to show the potential of the suggested solution methods, note that within this thesis we do not suggest one certain solution method. It was the intention to construct a powerful algorithm to solve stiff, possibly ill-conditioned nonlinear operator equations in general (see for example Figure 5.1). In our context this operator equation arises from a discretization of the Reynolds-averaged Navier-Stokes equations together with a boundary value problem formulated in Section 2.4.

The second topic going hand in hand with the discretization strategy is accuracy, in particular the assessment of accuracy toward the incompressible limit. Such a topic is of importance to find a closed representation of equations to model compressible flow phenomena.

Since strict mathematical analysis and proofs with respect to both accuracy assessment of the discretization schemes and convergence for the solution algorithms are missing, and moreover an overall solution theory does not exist for the analytic equations of interest, it is the goal to find a representative set of examples with varying flow complexity. We follow this idea considering examples illustrating different representative flow phenomena. These examples comprise flows around an airfoil at transonic inflow Mach numbers developing shocks, flows around wings and wing–body configurations at different inflow Mach numbers and flows around wing–body configurations at high angle of attack at low inflow Mach number. The last examples develop large separations as well as global incompressible together with strong locally compressible effects. The simulation of such flows often represents a particular challenge for a solution algorithm.

We close this introduction of this section with one important remark. During the

development of solution algorithms it figured out that unless an algorithm does not work reliably and robust for a large variety of parameters for basic flow cases under systematic increase of degrees of freedom, that is a systematic mesh refinement, it is not worthwhile to consider the method for large scale 3D flows. The 3D examples were in general considerably more difficult compared to the simulation of 2D flows.

6.1 Choice of suitable solution algorithm

Mentioned in Section 5.3.2, our general point of view on the construction of a solution algorithm has not considered the construction of several components. Therefore, let us shortly summarize the actual choice of several of these components, which have not been stated so far:

a) The nonlinear multigrid Algorithm 5.1.6:

   - The agglomeration strategy is determined in Section 5.1.2.
   - The projection is determined by (5.9), or equivalently (5.10).
   - The interpolation is determined by (5.12).

b) We need to determine the coefficients for (5.17). In all our computations we chose for both the mean flow and the turbulence flow equations the three stage scheme

$$\mathcal{A} := \begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{20} & 1 & 0 \\ 0 & \frac{2}{5} & 1 \end{pmatrix}, \quad b := \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{6.1}$$

   The choice (6.1) corresponds to recommendations given for example in [93, 91, 94].

c) Within this thesis we do not investigate the influence of the relaxation parameter ω_r introduced in (5.35) – (5.38). We only considered the choice $\omega_r = 1$.

d) Within this thesis we furthermore do not consider the smoothing techniques described in Theorem 5.3.2, 5.3.3, 5.3.4 and 5.3.5. In [45] and [90] it was shown both by example and analysis that these simplifications yield to severe stability problems of the algorithm in particular when mesh refinement studies are considered. In Section 7.2 such an analysis together with corresponding results and examples are given.

e) As smoother for Algorithm 5.1.6 we either consider Algorithm 3 or the method described in Theorem 5.3.1.

f) To determine the CFL number for iterate n, we follow (5.26). Treating the equations loosely coupled, we allow for different CLF numbers for the mean and turbulent flow equations,

$$\begin{aligned} \mathrm{CFL}_{\mathrm{mean}}(n) &= \min\left\{\mathrm{CFL}_{\mathrm{init}} \cdot f_{\mathrm{mean}}(n), \mathrm{CFL}_{\mathrm{mean,max}}\right\} && \text{(6.2a)} \\ \mathrm{CFL}_{\mathrm{turb}}(n) &= \min\left\{\mathrm{CFL}_{\mathrm{init}} \cdot f_{\mathrm{turb}}(n), \mathrm{CFL}_{\mathrm{turb,max}}\right\}, && \text{(6.2b)} \\ f_{\mathrm{mean}}(n) &= f_{\mathrm{turb}}(n) = \begin{cases} 1, & n < 10, \\ \gamma^{n-10}, & n \geq 10. \end{cases} && \text{(6.2c)} \end{aligned}$$

The parameters $\mathrm{CFL}_{\mathrm{init}}$, $\mathrm{CFL}_{\mathrm{mean,max}}$, $\mathrm{CFL}_{\mathrm{turb,max}}$ and γ are given in the description of the examples.

g) Treating the mean and turbulence flow equations loosely coupled allows for another important ingredient. In our implementation the multigrid algorithm for the turbulence flow equations does not work so reliably and efficiently compared with the mean flow equations. Therefore, we allow for doing more nonlinear iterations on the turbulence flow equations. For the Spalart-Allmaras model, in all test cases we used a relation of $1:5$ steps between the mean flow and turbulent flow equations, that is, for one multigrid cycle of the mean flow equations we performed five for the turbulent equation. For the $k\omega$-model an investigation with respect to this parameter choice is done in Section 6.11.2. For this model an adequate choice seems to be a significant severe problem compared with the Spalart-Allmaras model.

h) For 3D test cases it was often necessary to have a good initial guess. Therefore, we applied full multigrid. On coarse meshes our strategy was to apply 50 iterations (see for example Figure 6.8) on each level.

i) All test cases are computed fully turbulent that is without transition and in a parallel environment. The number of domains used is given as information in the tables describing the parameters of the test cases.

j) For several examples work units are shown. To compute the work units, we multiply the total CPU time for one computation with the number of domains used.

k) The application of GMRES Algorithm 5.27 was always done with the initial guess $\mathbf{x}^{(0)} = 0$. Though the exact construction of $\frac{d\mathbf{R}}{d\mathbf{W}}$ is available (see Chapter 4), its construction and application is a rather time consuming process. Hence, for large scale applications we have replaced it considering a symmetric finite difference to evaluate the matrix vector product,

$$\frac{d\mathbf{R}}{d\mathbf{W}}(\mathbf{W})\,\mathbf{h} \approx \frac{1}{2\varepsilon}\left(\mathbf{R}(\mathbf{W}+\varepsilon\mathbf{h}) - \mathbf{R}(\mathbf{W}-\varepsilon\mathbf{h})\right).$$

Now, finding a suitable ε is in general not straightforward and a further open problem. Some choice for ε may be found in the textbook of Blazek and by Nishikawa et al. [6, 68] or in [65]. For the examples given here the, simple choice $\varepsilon = 10^{-6}$ worked well.

ℓ) The threshold parameter required for the line search Algorithm 5.1.1 was in general chosen by parameter $\gamma_{\text{line}} = 10$. For an investigation of this parameter we refer to [43].

These principal parameter choices represent the basic practical guideline followed. In general it is observed that for a new test case an adaptation of the left-over parameters was necessary. In the examples considered in the following sections, we investigate for several test cases the influence of some of the parameters, to get an idea of their impact to the solution algorithm. Note, efficiency of the algorithms is for sure an important property. But the main focus is to be in a position to actually solve the boundary value problems formulated in Section 2.4. This task is already hard enough. And a method, which does not have the potential to solve these problems, is inefficient by construction.

6.2 Characteristic values

One important measure to evaluate the power and potential of considered solution algorithms is the history of residuals. For the mean flow equations we consider the density residual, which is evaluated by

$$\text{density residual}(n) \;:=\; \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{mean},\rho}\left(\mathbf{W}^{\mathrm{T_n}}\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}} \Big/ \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{mean},\rho}\left(\mathbf{W}_\infty\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}}.$$

For the one-equation Spalart-Allmaras model we denote the residual by,

$$\text{turbulent residual}(n) \;:=\; \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{turb}}\left(\tilde{\nu}^{\mathrm{T_n}}\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}} \Big/ \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{turb}}\left(\tilde{\nu}_\infty\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}},$$

and for the two-equation $k\omega$-model we evaluate both corresponding residuals

$$k-\text{residual}(n) \;:=\; \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{turb},k}\left(k^{\mathrm{T_n}},\omega^{\mathrm{T_n}}\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}} \Big/ \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{turb},k}\left(k_\infty,\omega_\infty\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}},$$

$$\omega-\text{residual}(n) \;:=\; \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{turb},\omega}\left(k^{\mathrm{T_n}},\omega^{\mathrm{T_n}}\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}} \Big/ \sqrt{\sum_{j=1}^{N_{elem}} \frac{\left(\mathbf{R}_{j,\text{turb},\omega}\left(k_\infty,\omega_\infty\right)\right)^2}{\left(\text{vol}(\Omega_j)\right)^2}}.$$

A computation is truncated and the result is accepted as soon as

$$\text{density residual}(n) < 10^{-14}. \tag{6.3}$$

The example considered in Section 6.8 presents the necessity for such a strict residual. Moreover, it is also a good indicator that an early stopping of the iteration after a residual reduction of only about 10^{-i}, $i = 2, \ldots 6$, which is often observed in applications, in general leads to questionable results. Early stopping introduces a further error component one cannot control.

Using free stream variables to normalize the residuals of the turbulence flow equations, one may not observe such a decrease of the residual. This is because the free stream is often not a good initial guess, and hence in the initial phase of the iteration an increase of the residual for these equations is observed. As an example we refer to Figures 6.58, 6.59 (left) and 6.63 (left), where the initial increase of the residual of the k-equation of several orders of magnitude can be observed for the initial iterations, before it starts to converge.

To investigate the examples we additionally use several in aerodynamics well established scalar values and distributions. Scalar values are the

a) drag coefficient C_D,

b) lift coefficient C_L.

These can be further subclassified by

$$C_D = C_{D,p} + C_{D,v}, \qquad C_L = C_{L,p} + C_{L,v},$$

where the subindex p denotes the contribution of forces corresponding to pressure, v the contribution corresponding to the viscous portion. Using polar coordinates

$$\begin{aligned} g(\alpha) &:= (0, \cos\alpha, 0, \sin\alpha, 0)^T, \\ h(\alpha) &:= (0, -\sin\alpha, 0, \cos\alpha, 0)^T, \end{aligned}$$

where α is the angle of attack, these are defined by

$$\begin{aligned} C_{L,p}(W) &:= \frac{2}{\rho_\infty u_\infty^2} \left\langle \int_{\partial D} \langle f_c(W), n\rangle \, ds(y), h(\alpha) \right\rangle, \\ C_{L,v}(W) &:= \frac{-2}{\rho_\infty u_\infty^2} \left\langle \int_{\partial D} \langle f_v(W), n\rangle \, ds(y), h(\alpha) \right\rangle, \end{aligned}$$

and

$$\begin{aligned} C_{D,p}(W) &:= \frac{2}{\rho_\infty u_\infty^2} \left\langle \int_{\partial D} \langle f_c(W), n\rangle \, ds(y), g(\alpha) \right\rangle, \\ C_{D,v}(W) &:= \frac{-2}{\rho_\infty u_\infty^2} \left\langle \int_{\partial D} \langle f_v(W), n\rangle \, ds(y), g(\alpha) \right\rangle. \end{aligned}$$

Evaluating these boundary integrals numerically in exactly the way used for the discretization, these values are consistently determined. Related to the evaluation of the force coefficients is the determination of surface pressure distribution C_p and surface skin friction C_f,

$$\begin{aligned} p\left(W\left(x\right)\right) &:= \left\langle \left\langle f_c\left(W\left(x\right)\right), n\left(x\right)\right\rangle, \left(0, n\left(x\right), 0\right)^T \right\rangle, \qquad x \in \partial D, \\ C_p\left(W\left(x\right)\right) &:= \frac{2}{\rho_\infty u_\infty^2}\left(p\left(W\left(x\right)\right) - p_\infty\right), \qquad x \in \partial D, \\ C_f\left(W\left(x\right)\right) &:= \frac{2}{\rho_\infty u_\infty^2}\left\langle \left\langle f_v\left(W\left(x\right)\right), n\left(x\right)\right\rangle, \left(0, t\left(x\right), 0\right)^T \right\rangle, \qquad x \in \partial D, \end{aligned}$$

which is implemented in a straightforward and consistent way by evaluation of the boundary flux. Note that $t = t(x)$ denotes a vector in the tangential space, which is not directly given. In the implementation considered the tangential vector is determined by

$$t(x) = \frac{1}{\|\tilde{t}(x)\|}\tilde{t}(x), \qquad \tilde{t}(x) = \left(\rho u\right)\left(x\right) - \left\langle \left(\rho u\right)\left(x\right), n\left(x\right)\right\rangle n\left(x\right), \qquad x \in \partial D.$$

For the assessment of inviscid flows we furthermore define the total pressure

$$\begin{aligned} p_{\mathrm{tot}}\left(W\left(x\right)\right) &= p\left(W\left(x\right)\right)\left(1 + \frac{1}{2}\left(\gamma - 1\right)M^2\right)^{\gamma/(\gamma-1)}, \\ p_{\mathrm{tot},\infty} &= p_\infty\left(1 + \frac{1}{2}\left(\gamma - 1\right)M_\infty^2\right)^{\gamma/(\gamma-1)}, \end{aligned}$$

and the total pressure loss

$$\text{Total pressure loss}\left(W\left(x\right)\right) := \frac{p_{\mathrm{tot},\infty} - p_{\mathrm{tot}}\left(W\left(x\right)\right)}{p_{\mathrm{tot},\infty}} = 1 - \frac{p_{\mathrm{tot}}\left(W\left(x\right)\right)}{p_{\mathrm{tot},\infty}}.$$

The value p_∞ is given in Section 2.3 and pressure p is determined by the equation of state (2.3). For inviscid incompressible flow the total pressure loss vanishes. Hence, this distribution is an in indicator for the accuracy of the numerical scheme.

6.3 Example 1: CASE 9, RAE 2822

The first example considered is a classical well known test case from the literature (see e.g. [13]) denoted by CASE 9. The relevant physical conditions are:

- Geometry: RAE 2822 airfoil
- Reynolds number: $Re = 6.5 \cdot 10^6$

	Coarse	Medium	Fine
Mesh size	320×64	640×128	1280×256
No. of quadrilaterals	20480	81920	327680
$\mathrm{CFL}_{\mathrm{init}}$	10	10	10
γ	5	5	5
$\mathrm{CFL}_{\mathrm{mean,max}}$	1000	1000	1000
$\mathrm{CFL}_{\mathrm{turb,max}}$	1000	1000	1000
Multigrid cycle (mean, turb)	4w, 4w	4w, 4w	4w, 4w
Number of domains	4	8	24
Work units: V1	1455	19056	207168
Work units: V2	1482	18240	193178
Work units: V3	840	10872	117528
Work units: V4	777	10200	107976

Table 6.1: Mesh data and parameters for the test case CASE 9, RAE2822

- Inflow Mach number: $M_\infty = 0.73$
- Angle of attack: AoA = 2.79 °.

Turbulence is modeled using the Spalart-Allmaras model (2.52). We perform the computations on a sequence of C-type structured meshes described in Table 6.1. A plot of the finest mesh in a neighborhood of the airfoil is given in Figure 6.1 (left). The outer boundary of the domain is located 20 chord lengths away from the airfoil. This test case represents a basic flow case which exhibits a shock at the upper surface of the airfoil. This can be observed in the $\mathrm{C_p}$ distribution in Figure 6.3 (left).
To approximate a solution of the corresponding set of algebraic equations (3.97), we apply the algorithm corresponding to Theorem 5.3.1 together with a variation in the stopping criterion and either (5.41) or (5.42). In detail, we investigate the four following variants:

- V1: Algorithm (5.41) is applied together with algorithm (5.36) using 5 symmetric sweeps.
- V2: Algorithm (5.41) is applied together with algorithm (5.36) using 3 symmetric sweeps.
- V3: Algorithm (5.42) is applied together with algorithm (5.36) using 5 symmetric sweeps.
- V4: Algorithm (5.42) is applied together with algorithm (5.36) using 3 symmetric sweeps.

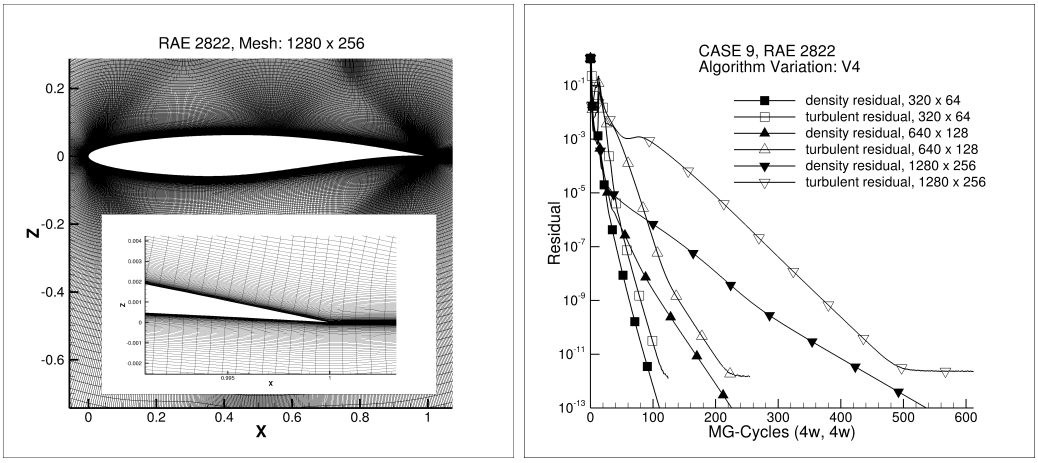

Figure 6.1: Left: Mesh of RAE 2822, 1280×256, right: Convergence history CASE 9, RAE 2822

Remark 6.3.1 *Freezing the preconditioner means not only that the block sparse matrix is just constructed on the first stage, but also the block LU-decomposition required in (5.36) to evaluate* $\mathbf{Tri}_{L_i}^{-1}$ *is only computed on the first stage.*

The relaxation parameter in (5.29) and (5.30) is chosen by $\varepsilon = 1/2$. Table 6.1 summarizes further chosen algorithmic parameters as well as work units required until the iteration was stopped by the convergence criterion (6.3).

The results obtained suggest that for this test case the number of sweeps, either 3 or 5, has only minor impact on the total computational time. But a view on the work units shows that these can be reduced by about a factor of two, when comparing variant V1 with V3 and V2 with V4. A loss of robustness or any other drawback of (5.42) compared with (5.41) was not observed.

The convergence histories of the drag and lift coefficients are shown in Figure 6.2 (left). The work units are given in Table 6.1, and a plot of work units over number of mesh points is shown in Figure 6.2 (right). It can be observed that we have a scaling that is roughly in between $O(N)$ and $O(N^2)$.

Figure 6.3 shows the C_p and C_f distributions compared with experimental data given in [13], which are in good agreement. Unfortunately, trying to increase the depth of the multigrid cycle, in particular for the finest computational mesh, failed. These observations are indicators that further research in multigrid, and in particular the construction of coarse grid levels is required. Nevertheless, in a first step a much better understanding for the interaction of all these several algorithmical components is needed.

Remark 6.3.2 *Freezing the preconditioner on the first stage seems to be an appropriate approach to save computational time for an overall more efficient method.*

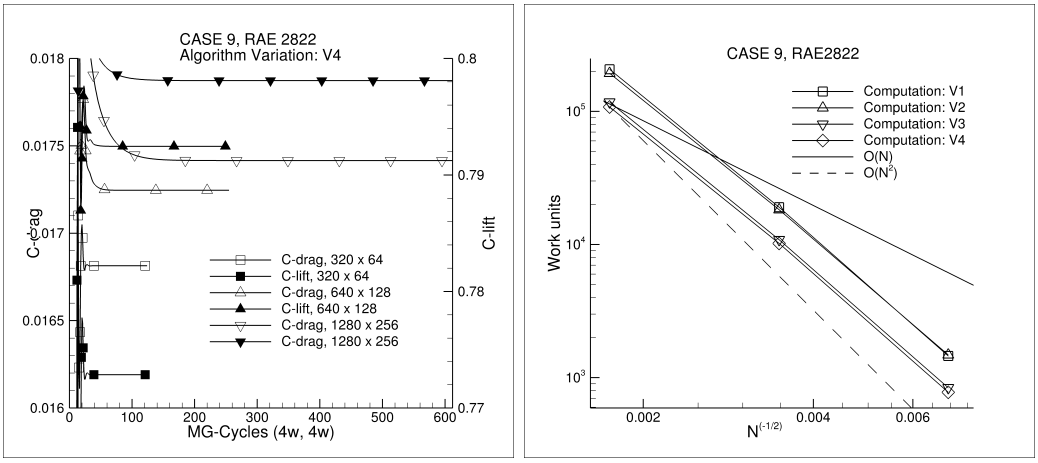

Figure 6.2: Left: Convergence history for lift coefficient and drag coefficient, right: Work units for RAE 2822

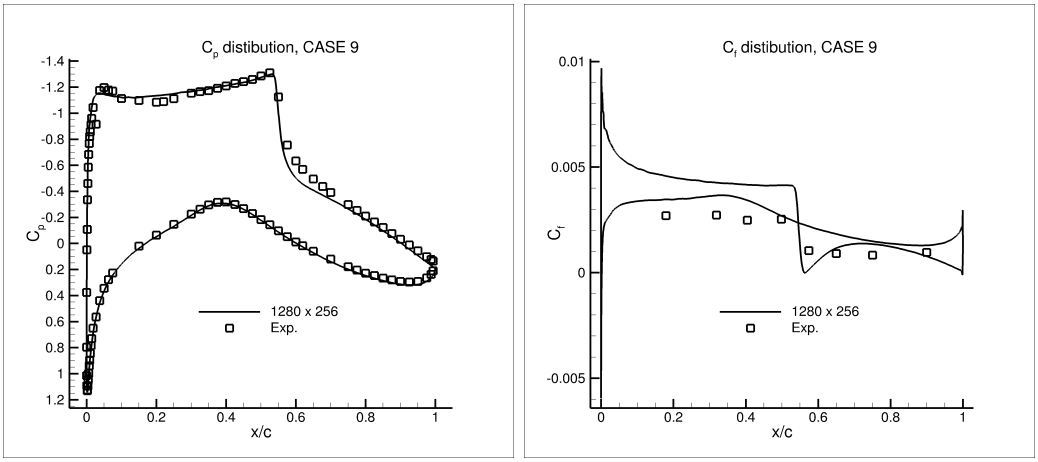

Figure 6.3: Left: C_p distribution for CASE 9, RAE 2822, right: C_f distribution

6.4 Example 2: MDA30P30N

To confirm the observation and conclusion of Section 6.3 we consider the high-lift configuration MDA30P30N. Compared to Section 6.3 this example represents other flow effects. It is a multi-element airfoil at high angle of attack characterized by incompressible and strong local compressible effects and regions of large separation. The meshes were generated by the SOLAR hybrid mesh generator [53]. To this end they are unstructured but consist of purely quadrilateral elements. A plot of the

	Coarse	Medium	Fine
No. of points	119510	240955	485832
lift coefficient	4.089312	4.122349	4.136353
drag coefficient	5.213078e-02	5.129756e-02	5.080194e-02
$\mathrm{CFL}_{\mathrm{init}}$	10	10	10
γ	5	5	5
$\mathrm{CFL}_{\mathrm{mean,max}}$	1000	1000	1000
$\mathrm{CFL}_{\mathrm{turb,max}}$	1000	1000	1000
Multigrid cycle (mean, turb)	4w, 3v	4w, 3v	4w, 3v
Number of domains	24	24	24
Work units: V1	74328	158136	423192
Work units: V3	40824	84528	269760

Table 6.2: Data for the test case MDA30P30N

finest mesh is given in Figure 6.4 (left). The physical parameters of the test case are as follows (see [37]):

- Geometry: MDA30P30N airfoil
- Reynolds number: $Re = 9.0 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.2$
- Angle of attack: AoA = 16.0 °.

Turbulence is modeled using the Spalart-Allmaras model (2.52). Though representing a high lift test case vortex correction was not used. Instead the farfield boundary is located 100 chord lengths away from the airfoil. The choice of the parameter settings, the number of points as well as computed lift and drag coefficients are given in Table 6.2. The relaxation parameter in (5.29) and (5.30) is chosen by $\varepsilon = 1/2$. Due to the negligible effect on the number of sweeps in the total computational time observed in the example of Section 6.3, only a fixed number of sweeps, namely 5 together with either (5.41) or (5.42) is investigated. A comparison of the work units given in Table 6.2 confirms the observation that freezing the preconditioner on the first Runge-Kutta stage gives roughly a speed-up of a factor two while an impact on the robustness of the solution method cannot be observed.
Convergence of the residual and the force coefficients is shown in Figure 6.4 (right) and Figure 6.5 (left). Work units are plotted in in Figure 6.5 (right). Linear scalability $O(N)$ of the algorithm shown is observed both for frozen and non-frozen preconditioner. With increasing mesh density the approximate linear scaling of $O(N)$ deteriorates.
Finally, a plot of the $\mathrm{C_p}$ and $\mathrm{C_f}$ distributions and comparison with experimental data available from [37] is given in Figure 6.6. These are in good agreement.

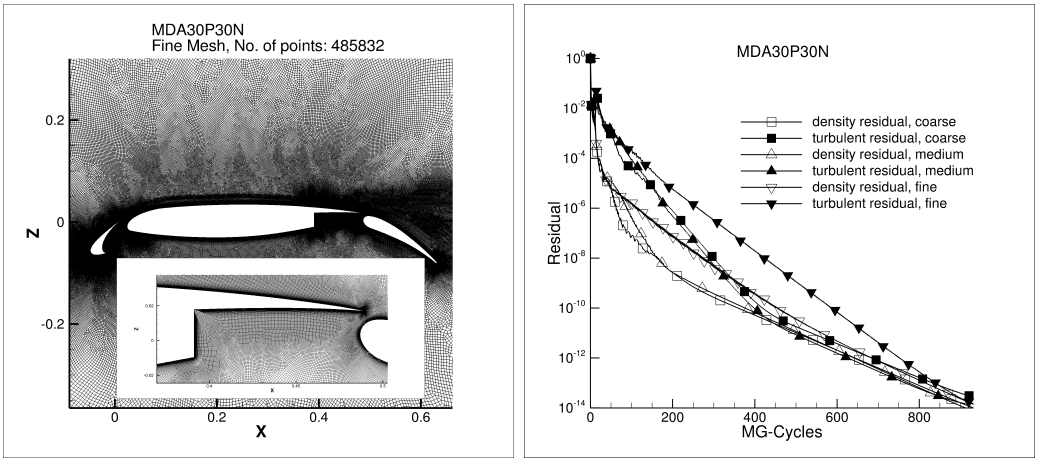

Figure 6.4: Left: Fine mesh of MDA30P30N, right: Convergence history

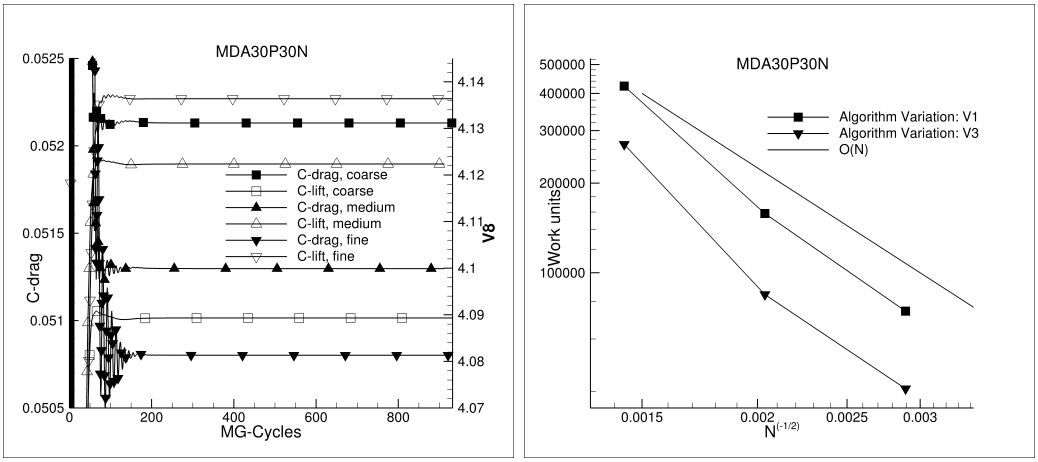

Figure 6.5: Left: Convergence history for lift and drag coefficient, right: Work units for MDA30P30N

Remark 6.4.1 *Having settled one major contributing factor to the total computational time, namely the construction of the preconditioner* **Prec***, we now consider the frozen variant (5.42) as standard. This puts us into the position to consider 3D test cases, which require inherently much more computational resources.*

6.5 Example 3: DPW III, Case 2, Wing 1

As an initial 3D test case we consider flow over a wing originally considered at the third Drag Prediction Workshop.
Data describing the meshes and parameters are given in Table 6.3. A plot of the

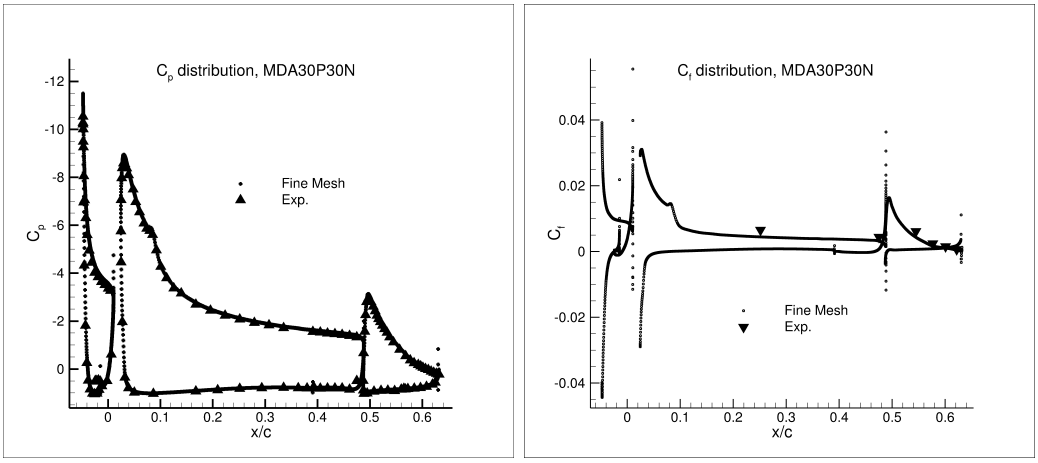

Figure 6.6: Left: C_p distribution for MDA30P30N, right: C_f distribution

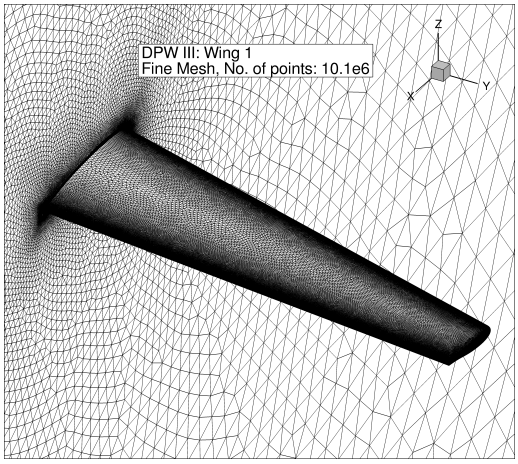

Figure 6.7: Fine mesh of DPW III Wing 1

finest surface mesh is given in Figure 6.7. The relevant physical conditions of the flow are

- Geometry: Drag Prediction Workshop III, Case 2, Wing 1
- Reynolds number: $Re = 5.0 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.76$
- Angle of attack: AoA = 0.5 °.

Turbulence is modeled using the Spalart-Allmaras model (2.52). We use this test case to investigate the influence of the relaxation parameter ε introduced in (5.29)

	Coarse	Medium	Fine
No. of points	2174364	5288507	10150588
No. of Tetrahedra	6710084	10672069	15052013
No. of Prisms	1976700	6784770	14931077
No. of Pyramids	0	0	115
No. of surface triangles	162314	297914	454484
No. of surface quadrilaterals	2820	6090	11240
drag coefficient(p)	1.494445e-02	1.466697e-02	1.470134e-02
drag coefficient(v)	6.063326e-03	6.101706e-03	6.106965e-03
drag coefficient	2.10077e-02	2.076867e-02	2.080831e-02
lift coefficient	4.730909e-01	4.765350e-01	4.783592e-01
$\mathrm{CFL}_{\mathrm{init}}$	10	10	10
γ	1.5	1.5	1.5
$\mathrm{CFL}_{\mathrm{mean,max}}$	1000	1000	1000
$\mathrm{CFL}_{\mathrm{turb,max}}$	1000	1000	1000
Multigrid cycle (mean, turb)	4w, 3v	4w, 3v	4w, 3v
Number of domains	96	120	180
Work units: V3, $\varepsilon = 1$	715968	1513200	2916000
Work units: V3, $\varepsilon = 7/10$	614016	1299240	2627280

Table 6.3: Mesh data for the test case DPW III: Case 2, Wing 1

and (5.30). In general, from our experience for large scale 3D flows, over-relaxation is not stable. For the wing flow the relaxation parameter could be reduced on all three meshes considered from $\varepsilon = 1$ to $\varepsilon = \frac{7}{10}$. Work units given in Table 6.3 show that the computational time can be reduced by 10% – 15% by the reduction of ε. The reason is that the number of multigrid cycles required to hit the truncation criterion (6.3) using over-relaxation could be reduced by about 10% on all three meshes. The convergence history is shown in Figure 6.8 for $\varepsilon = 7/10$ (right) and for $\varepsilon = 1$ (left).

Remark 6.5.1 *From this investigation we conclude that the over-relaxation parameter ε used in (5.29) and (5.30) has impact on the convergence. Observed that the gain in efficiency choosing $\varepsilon < 1$ for this and many other 3D flow cases is negligible compared to the influence of other parameters (e.g. multigrid cycle), we conclude that choosing $\varepsilon < 1$ is not worthwhile to consider. In particular not when one considers the induced instabilities to the algorithm in particular for 3D test cases. Hence, for all following 3D test cases we choose $\varepsilon = 1$.*

To approximate a good initial guess, a full multigrid approach has been followed. This is indicated by the minus numbering in the plots of Figure 6.8. Starting on the

coarsest considered level, an initial guess for the next finer mesh is approximated. In particular for 3D flows it turned out that full multigrid is often a necessary approach for the finding of a good initial guess, stabilizing significantly the whole solution algorithm.

The convergence history of the force coefficients given in Figure 6.9 (left) shows that on all three meshes roughly 50 multigrid cycles are required to reach stable lift and drag coefficients. Figure 6.9 (right) shows the required work units. A plot of C_p and C_f distribution at the 55% span location is given in Figure 6.10.

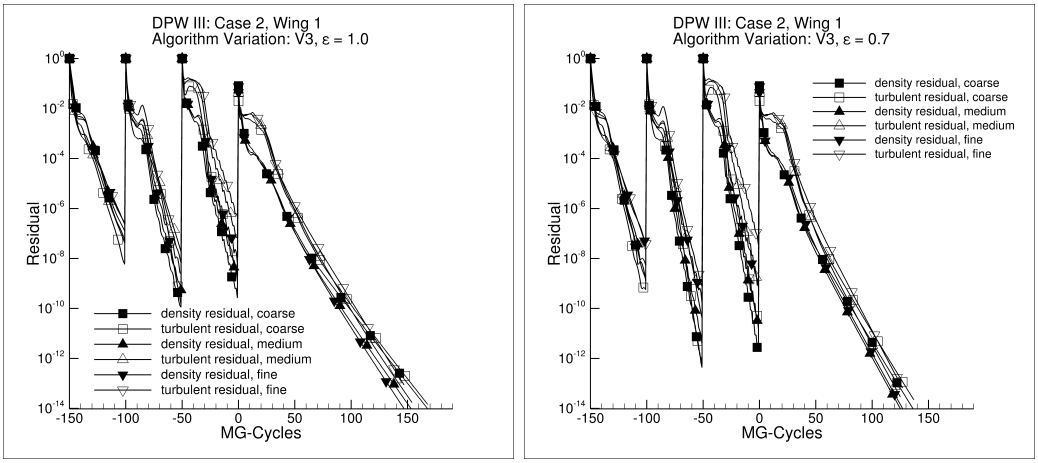

Figure 6.8: Convergence history for DPW III, Case 2, Wing 1 and $\varepsilon = 1$ (left) and $\varepsilon = \frac{7}{10}$ (right)

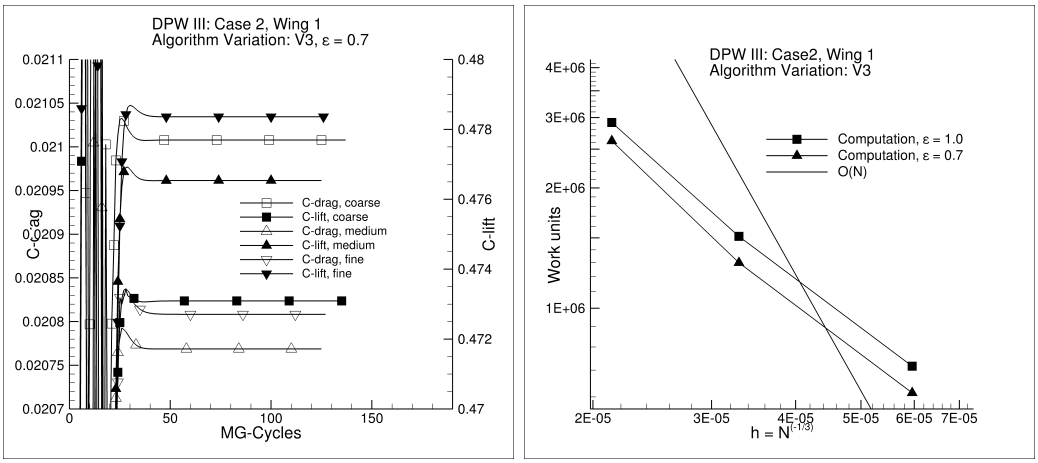

Figure 6.9: Left: Convergence history for DPW III of lift and drag coefficient, right: Work units for DPW III, Wing 1

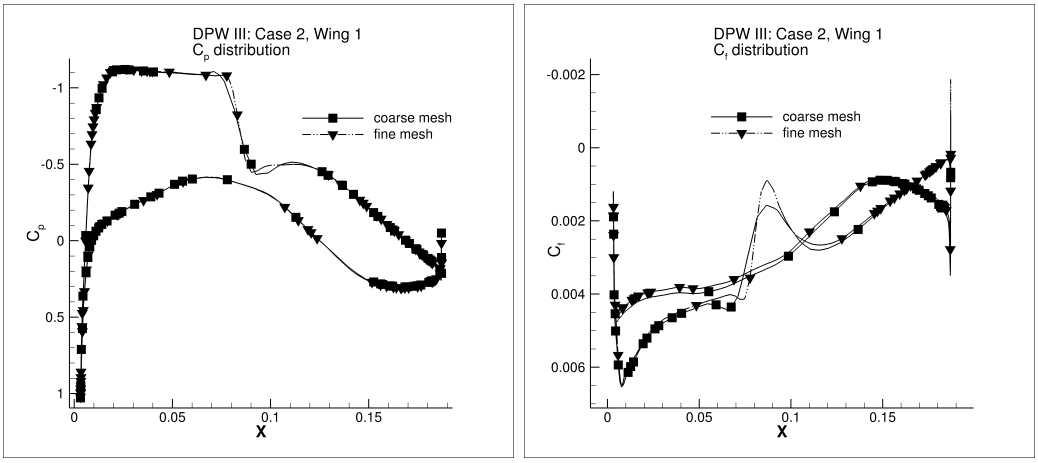

Figure 6.10: Computed C_p (left) and C_f (right) distribution for coarse and fine mesh at the 55% span location

6.6 Example 4: Transonic turbulent flow over a common research model

	Hybrid meshes		Hexahedral meshes	
Level	No. of Tetrahedra	No. of Prisms	No. of Hexahedrons	No. of points
L1	2555904	425984	638976	660177
L2	8626176	1437696	2156544	2204089
L3	20766720	3301376	5111808	5196193
L4	69728256	11261952	17252352	17441905
L5	166133760	26411008	40894464	41231169

Table 6.4: Mesh data for DPW5 CRM

As a further test case we consider a sequence of meshes representing a wing body configuration. We use this test case for several investigations. This test case was considered at the fifth AIAA Drag Prediction Workshop. The considered original meshes are block-structured. A sequence of hybrid meshes were generated from the pure hexahedral meshes. A detailed description of the meshes can be found in [110]. A plot of the L4 surface mesh is given in Figure 6.14 (left). Here we use meshes described in Table 6.4. For the sequence of meshes we searched for the angle of attack such that a target lift coefficient of $C_L = 0.5$ is reached. The target lift computation was stopped as soon as the lift coefficient is in the interval $[0.4999, 0.5001]$. The relevant physical conditions are:

- Geometry: Wing-body configuration, fifth AIAA Drag Prediction Workshop

- Reynolds number: $Re = 5.0 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.85$
- Target $C_L = 0.5$.

Turbulence is modeled using the Spalart-Allmaras model (2.52). The algorithm parameters are given in Tables 6.5, 6.6 and 6.7. We consider here only the convergence histories for the final angle of attack.

6.6.1 Influence of choice of gradients

Level	L1	L2	L3
drag coefficient(p)	$1.4164 \cdot 10^{-2}$	$1.3391 \cdot 10^{-2}$	$1.3253 \cdot 10^{-2}$
drag coefficient(v)	$1.0789 \cdot 10^{-2}$	$1.0968 \cdot 10^{-2}$	$1.1024 \cdot 10^{-2}$
drag coefficient	$2.4953 \cdot 10^{-2}$	$2.4359 \cdot 10^{-2}$	$2.4277 \cdot 10^{-2}$
AoA	2.27°	2.1484°	2.12°
CFL_{init}	3	3	3
γ	1.1	1.1	1.1
$CFL_{mean,max}$	1000	1000	1000
$CFL_{turb,max}$	1000	1000	1000
Multigrid cycle (mean, turb)	4w, 3v	4w, 3v	4w, 3v
Number of domains	24	48	72

Table 6.5: DPW5 CRM: Parameters and computed forces for hexahedral meshes for target lift coefficient $C_L = 0.5$ using TSL gradients (3.64)

First of all, we investigate the accuracy of the discretization with respect to the different gradient approximations (3.64) and (3.63) required for the viscous and source terms. Using gradient approximation (3.64) gives agreement of the discretization and the derivative terms used in the preconditioner. On the other hand, comparing results given for hexahedral meshes L1, L2 and L3 shown in Figure 6.11, it is observed that the convergence histories of (3.64) and (3.63) are similar. However, when comparing the results for gradient approximations (3.64) in Table 6.5 with results given in [50] we notice that the drag coefficient and angle of attack are slightly too low.

This test case demonstrates that the systematic error introduced using TSL gradients (3.64) (see Lemma 3.4.8) can have impact on the determined approximate solution yielding deviations in drag and lift coefficients. Hence, using such approximations should be handled with care.

Therefore, at least for the mean flow equations we always use gradient approximation (3.63). Then the determined angle of attack and drag coefficient given in

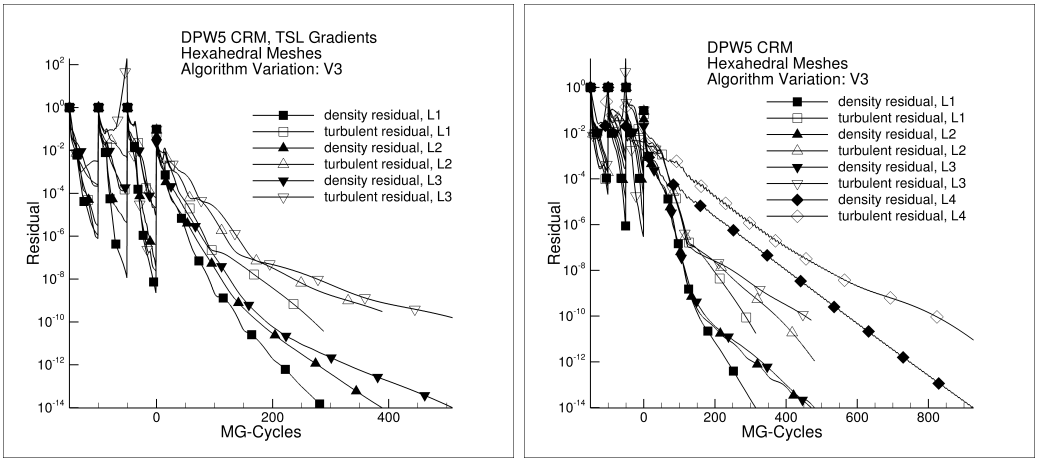

Figure 6.11: DPW5 CRM, left: Convergence history using (3.64) in the discretization, right: Convergence history using (3.63) in the discretization

Tables 6.6 and 6.7 for both the hexahedral and hybrid meshes are in good agreement with results shown in [50]. A plot of the C_p distribution for the hexahedral L1 and L3 meshes is shown in Figure 6.13 (right).

6.6.2 Assessment of low cost smoother

Level	L1	L2	L3	L4	L5
drag coefficient (p)	$1.4655 \cdot 10^{-2}$	$1.3673 \cdot 10^{-2}$	$1.3483 \cdot 10^{-2}$	$1.3365 \cdot 10^{-2}$	$1.3330 \cdot 10^{-2}$
drag coefficient (v)	$1.0677 \cdot 10^{-2}$	$1.1134 \cdot 10^{-2}$	$1.1302 \cdot 10^{-2}$	$1.1431 \cdot 10^{-2}$	$1.1478 \cdot 10^{-2}$
drag coefficient	$2.5332 \cdot 10^{-2}$	$2.4807 \cdot 10^{-2}$	$2.4785 \cdot 10^{-2}$	$2.4796 \cdot 10^{-2}$	$2.4808 \cdot 10^{-2}$
AoA	2.364°	2.193°	2.158°	2.133°	2.1245°
CFL_{init}	3	3	3	2	2
γ	1.1	1.1	1.1	1.05	1.05
$CFL_{mean,max}$	1000	1000	1000	250	50
$CFL_{turb,max}$	1000	1000	1000	250	50
Multigrid cycle (mean, turb)	4w, 3v	4w, 3v	4w, 3v	3v, 2v	3v, 2v
No. of domains	24	48	72	192	384

Table 6.6: DPW5 CRM: Parameters and computed forces for hexahedral meshes for target lift coefficient $C_L = 0.5$

So far, for all test cases considered we used a simplified "low cost smoother" described in Theorem 5.3.1. Turning now to mesh refinement studies for 3D flow simulations, it is the intention to show that these simplifications may yield a significant loss in robustness of the solution methods.

Noted in Tables 6.6 and 6.7 for the hybrid L5 mesh it was necessary to reduce the CFL number to 250, and for the hexahedral L4 and L5 mesh we had to reduce the CFL number to 250 and 50 resp. Additionally, let us comment on the convergence

Level	L1	L2	L3	L4	L5
drag coefficient(p)	$1.6112 \cdot 10^{-2}$	$1.4335 \cdot 10^{-2}$	$1.3792 \cdot 10^{-2}$	$1.3531 \cdot 10^{-2}$	$1.3421 \cdot 10^{-2}$
drag coefficient(v)	$1.0333 \cdot 10^{-2}$	$1.0830 \cdot 10^{-2}$	$1.0964 \cdot 10^{-2}$	$1.1236 \cdot 10^{-2}$	$1.1330 \cdot 10^{-2}$
drag coefficient	$2.6445 \cdot 10^{-2}$	$2.5165 \cdot 10^{-2}$	$2.4756 \cdot 10^{-2}$	$2.4767 \cdot 10^{-2}$	$2.4751 \cdot 10^{-2}$
AoA	2.314°	2.1892°	2.168°	2.1355°	2.127°
$\mathrm{CFL}_{\mathrm{init}}$	3	3	3	3	2
γ	1.1	1.1	1.1	1.1	1.05
$\mathrm{CFL}_{\mathrm{mean,max}}$	1000	1000	200	1000	250
$\mathrm{CFL}_{\mathrm{turb,max}}$	1000	1000	50	1000	250
Mult. cycle (mean, turb)	4w, 3v	4w, 3v	4w, 3v	4w, 3v	3v, 2v
Number of domains	24	48	72	192	384

Table 6.7: DPW5 CRM: Parameters and computed forces for hybrid meshes for target lift coefficient $C_L = 0.5$

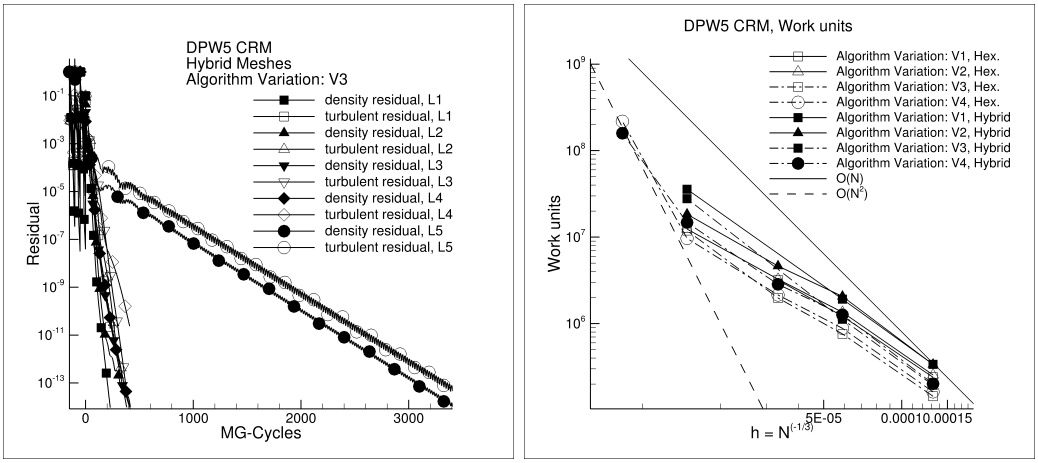

Figure 6.12: DPW5 CRM, left: Convergence history for hybrid meshes, right: Work units for variation of methods and grids

history shown for the hexahedral L5 mesh given in Figure 6.13 (left). Unfortunately, only after about 4000 multigrid cycles it was possible to increase the CFL number manually up to 50. So, the process of this computation cannot be interpreted as reliable. And, additionally, an approach to find a maximum CFL number is required. It can be doubted that such a method exists in general (see Section 5.1.7 for a more detailed discussion). An even better way out is to find a way to remove the strong sensitivity with respect to the final CFL number. Moreover, the reduction of the CFL numbers explain the loss of scaling of the algorithms indicated in Figure 6.12 (right), where the complexity of different variations of the algorithms are considered.

Quite obviously, the complexity of the algorithm varies in between $O(N)$ and $O(N^2)$ except for the hybrid L5 mesh, shown in Figure 6.12 (right). For this mesh both a reduction of the CFL number and a reduction of the multigrid cycle were necessary

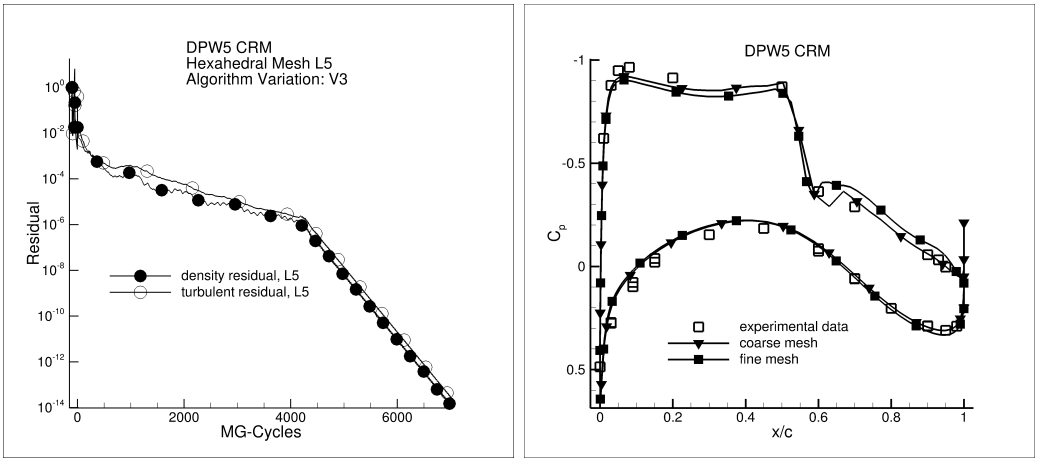

Figure 6.13: DPW5 CRM, left: Convergence history for L5 hexahedral mesh, right: Computed C_p distribution for hexahedral L1 and L3 mesh at the 50.2% span location

to converge this test case 14 orders of magnitude. This explains the tremendous increase in complexity for the L5 mesh. Furthermore, it is observed that for this test case the loss of robustness is more severe for the hexahedral meshes. This observation, in particular the loss of robustness of multigrid, goes along with other results presented in the literature (see [69]). Nevertheless, at least it was possible to converge both sequences of meshes given in Table 6.4 to machine accuracy using an agglomeration multigrid together with a smoother described in Theorem 5.3.1.

We conclude, considering mesh refinement studies in 3D increasing systematically the number of degrees of freedom yielding more accurate solutions, in general may lead to both a loss of scalability and robustness of the algorithm suggested in Theorem 5.3.1.

We use this test case to show, that one possible way to overcome the loss in robustness is to avoid simplifications of the smoother.

6.6.3 Assessment of Newton kind smoother

Instead of considering a severe simplification of the smoother, we now apply directly the Implicit Runge-Kutta smoother given by Algorithm 3. For simplicity we denote this smoother as Newton-GMRES, though being aware of the fact that it represents some kind of damped or regularized Newton method. We truncate the GMRES Algorithm 5.2.1 after 20 iterations. To determine (6.2a) and (6.2b) we chose

$$\mathrm{CFL}_{\mathrm{init}} = 3, \quad \mathrm{CFL}_{\mathrm{mean,max}} = 1000, \quad \mathrm{CFL}_{\mathrm{turb,max}} = 1000, \qquad \gamma = \frac{3}{2}.$$

The convergence history is shown in Figure 6.14 (right). Hence, eliminating the simplification introduced by Theorem 5.3.1 it is possible to reach higher levels of CFL numbers indicating an improvement of the robustness of the overall solution strategy.

It is not unexpected that simplifications in a solution algorithm moving it further away from Newton's method can have rather serious detrimental effects on the convergence and robustness. These effects may often only be observed with respect to mesh refinement studies for 3D test cases. Hence, this is a suitable example demonstrating such an effect. Being aware of such a behavior it is necessary to have these kinds of Newton algorithms available in a computer code approximating solutions of the RANS equations. It can be assumed that with an increase of mesh size, that is an increase of the number of degrees of freedom, such behavior is reproducible for many other examples.

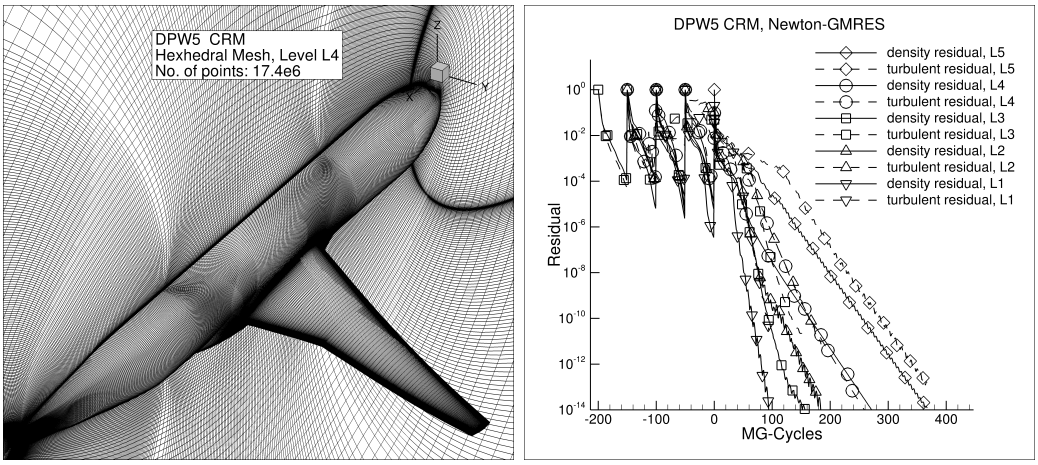

Figure 6.14: Left: Hexahedral L4 mesh of DPW5 CRM Right: Convergence history of using Algorithm 3 as smoother

6.7 Example 5: High-lift Prediction Workshop II, Case 2a

To confirm the necessity of regularized Newton kind algorithms, we consider a test case from the second High-lift Prediction Workshop. High Reynolds number turbulent flows at low Mach number and high angle of attack are of particular importance. An aircraft in landing configuration is a typical example of such a simulation scenario. For compressible flow solvers the simulation of such flows is a challenge in itself. On the one hand often large regions of separations and vortices can be ob-

served, on the other hand the mixture of compressible and incompressible effects make the simulation of such flows even harder. The relevant physical conditions are

- Geometry: Case 2a, High-lift Prediction Workshop II
- Reynolds number: $Re = 1.35 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.175$
- Angle of attack: AoA = 7.0 °.

Turbulence is modeled using the Spalart-Allmaras model (2.52). The number of degrees of freedom are about $10.2 \cdot 10^6$. It is often observed that for these flow simulations it is very hard to obtain a converged (machine accurate) result [83, 82].

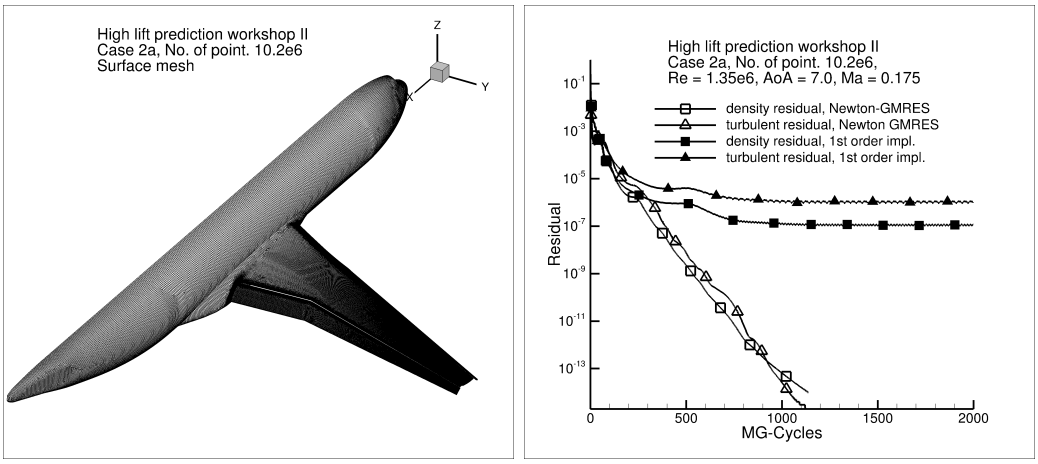

Figure 6.15: Left: Surface mesh, right: Convergence history

To conclude that there does not exist a steady state seems often to be short sighted. We give here an example where at least within in our code framework, only a smoother based on Newton's method was able to find the steady state solution.

Figure 6.15 (left) shows the surface mesh, and Figure 6.15 (right) the convergence histories for two different solution algorithms. The first order preconditioned Runge-Kutta smoother (see Theorem 5.3.1) made the simplification that the number of Krylov steps is zero. Hence, algorithm (5.41) is used to approximate the steady state solution. The corresponding convergence history shown in Figure 6.15 (right) is the one with the filled symbols. The consequence of this simplification of the nonlinear multigrid smoother is that convergence could not be obtained, though trying several parameter choices such as different CFL numbers. Finally, we must admit that within our implementation it was not possible to reach a machine accurate solution using algorithm (5.41).

Working with a method where the number of Krylov steps was set to 20, that is the nonlinear multigrid smoother Algorithm 3 is used, without major issues and a CFL number of 250 a fully converged solution could be obtained. This is shown in Figure 6.15 (right). The corresponding convergence history is the one with the non-filled symbols. Hence, this example presents again the importance to have the full scope of methods to solve nonlinear problems available.

6.8 Example 6: 3D three element high lift wing-body configuration

No. of points	10733766
No. of Tetrahedra	11590308
No. of Prisms	17128338
No. of Pyramids	1523
No. of surface triangles	628690
No. of surface quadrilaterals	23933
$\mathrm{CFL}_{\mathrm{init}}$	10
γ	1.5
$\mathrm{CFL}_{\mathrm{mean,max}}$	250
$\mathrm{CFL}_{\mathrm{turb,max}}$	50
Multigrid cycle (mean, turb)	3v, 2v
Number of domains	192

Table 6.8: 3D three element wing-body configuration: Mesh data and parameters

One may argue that the necessity to obtain fully converged result is more from theoretical nature than a practical one. On the other hand, it is obvious that a none converged solution has an unknown error component with an unknown impact on the relevant physical data. To demonstrate, in which way such an error component can influence data of interest, is the goal of this example.

We consider a 3D high lift configuration. The computational grid available as well as the parameter setting are described in Table 6.8. The mesh was generated with CENTAUR [29]. A visualization from the top and front of the computational mesh is given in Figure 6.16. The relevant physical conditions are

- Geometry: 3D high lift wing-body configuration
- Reynolds number: $Re = 15.495 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.1816$

- Angle of attack: AoA = 24.0 °.

Turbulence is modeled using the Spalart-Allmaras model (2.52). At the 50% span location a plot of the C_p and C_f distributions is given in Figure 6.17 and the convergence history is shown in Figure 6.18.

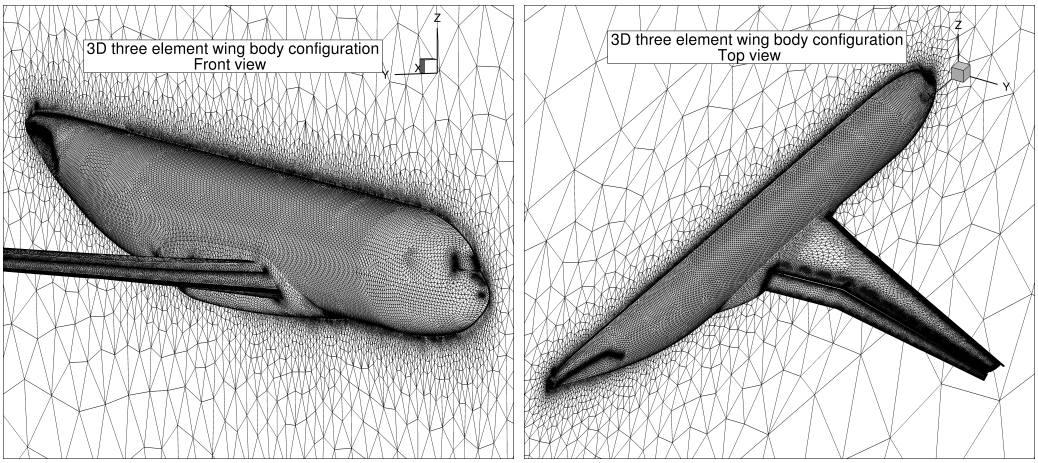

Figure 6.16: Mesh of high lift 3D three element wing-body configuration in front view (left) and top view (right)

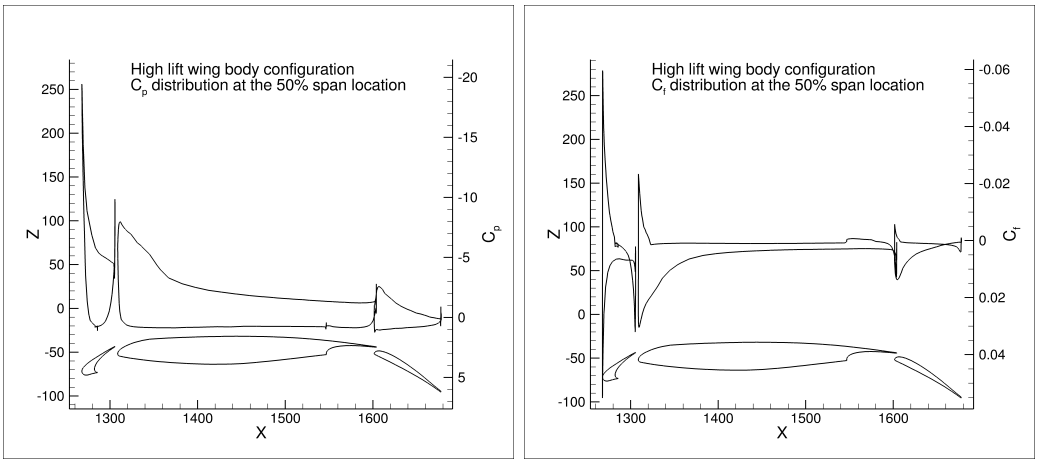

Figure 6.17: Computed C_p (left) and C_f (right) distribution of high lift 3D three element wing-body configuration at the 50% span location

In the convergence history shown in Figure 6.18 there is substantial slowdown of the residual decrease after an initial drop of the residual of about 6 orders of magnitude. It takes about 1000 multigrid cycles until the residual drops again with its asymptotic convergence rate. Though it is unknown what causes this behavior of

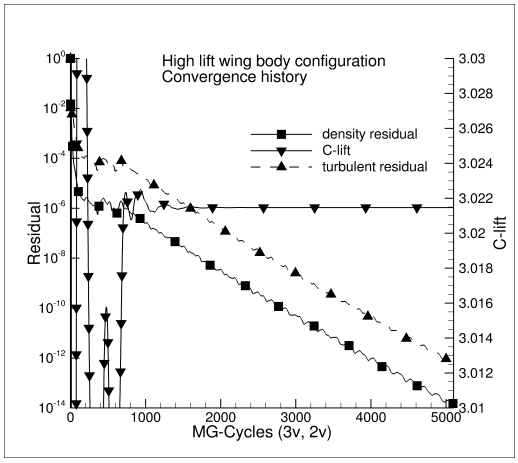

Figure 6.18: Convergence history for high lift 3D three element wing-body configuration

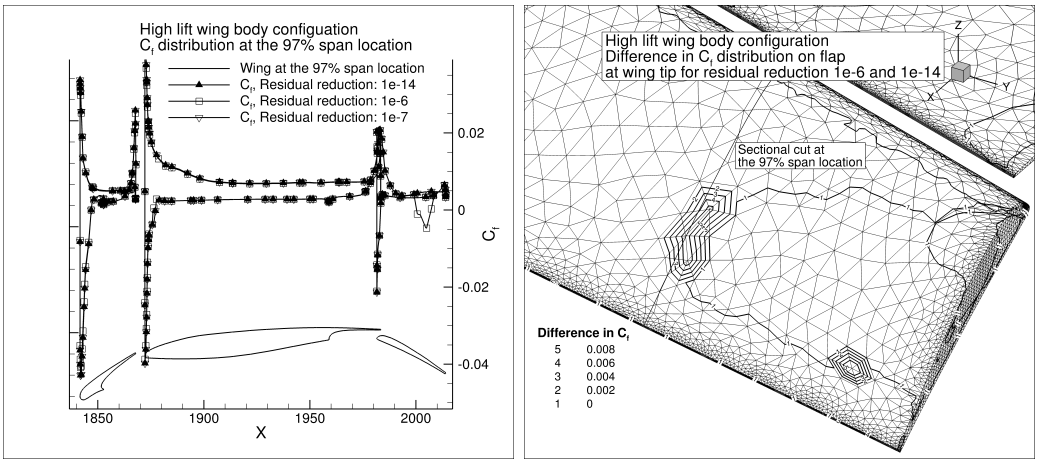

Figure 6.19: Computed C_f distribution (left) of high lift 3D three element wing-body configuration at the 97% span location and difference of C_f near wing tip for different residual reductions

the solution algorithm, it can be assumed that certain flow features described by equations (2.1a) and (2.18) have not developed after the initial drop of the residual of about 6 orders of magnitude. As a confirmation of this assumption a plot of the C_f distribution at the 97% span location is given in Figure 6.19 (left). Here the C_f distribution before the slowdown (residual drop of about 6 orders), after the slowdown (residual drop of about 7 orders) and for the final solution (residual drop of about 14 orders) is shown. Whereas the C_f distribution after the slowdown and for the final solution do not show major differences, there is a significant difference between the final solution and the approximate solution before the slowdown. As

can be seen by Figure 6.19 (right) the size of the separation on the flap at the wing tip is smaller for the final solution when compared with the approximate solution after a residual reduction of about 6 orders. As soon as the final convergence rate is reached, the difference between the approximate solution and final solution seems to be negligible. Hence, it can be assumed that for this experiment an adequate approximate solution is given for a residual reduction of about 7 orders of magnitude. However, this conclusion can only be made because we have knowledge about the full convergence history and final solution. It is clear that premature stopping of the iteration may yield flow predictions significantly differing from the final solution of boundary value problems corresponding to equations (2.1a) and (2.18) with respect to the discretization considered in this thesis.

6.9 Investigations for the incompressible limit

It is not only the goal of this Section to investigate the modifications presented in Section 3.3, but, additionally, to investigate the accuracy of the different discretization schemes. Unfortunately, analytic solutions are not available for any of the examples considered. In this regard, there is no objective argument for evaluating the accuracy of the different discretization strategies. Therefore, at the outset, we are not in a position to conclude that one discretization strategy yields a more accurate result than another one. Naturally, when possible we evaluate accuracy with respect to experience and expectations.

On the other hand, to give some objective guidance, we extrapolate from the computed data a data point corresponding to an infinitely fine mesh together with an approximate order of the method. For this purpose we solve the following system of equations

$$f_i(c_\infty, \kappa, \omega) = \|c_\infty - c_{i,\text{comp.}}\| - \kappa \left(\frac{1}{(\text{nDOF}_i)^{1/\text{dim}}} \right)^{\omega}, \qquad i = 1, 2, 3. \tag{6.4}$$

Here, dim denotes the space dimension, and the term $1/\left(\text{nDOF}\right)^{1/\text{dim}}$ corresponds to the mesh size; that is, we assume that the mesh size behaves like

$$h \sim 1/\left(\text{nDOF}\right)^{1/\text{dim}}, \tag{6.5}$$

and nDOF denotes the number of degrees of freedom. For a method of order ω in the asymptotic regime the estimate

$$\|c_\infty - c_{\text{comp.}}\| \leq \kappa \cdot h^{\omega}, \qquad \kappa \geq 0, \tag{6.6}$$

holds true. For the unknowns c_∞, κ and ω we reformulate estimate (6.6) as the system of three equations (6.4). Having three values of c_i, $i = 1, 2, 3$, for a given

sequence of three meshes satisfying that the mesh sizes behave as

$$h_i \sim 1/\left(\mathrm{nDOF}_i\right)^{1/\mathrm{dim}},$$

we need to solve for a root of system (6.4) to determine the three unknowns. This analysis was done for the NACA 0012 airfoil test case, and for the transonic turbulent flow over the Common Research Model. The computed results for the extrapolated force coefficient c_∞ as well as the corresponding order of convergence ω and constant κ on an infinite mesh can be found in the Tables 6.10 and 6.12.

Before for these test cases an analysis based on (6.4) is performed, a short consideration about the expected order of convergence ω is required. Note the following facts:

a) The order of the discretization of the convective terms can only be shown to be of second order under special circumstances, for example a Cartesian mesh.

b) Gradients used to discretize the viscous terms are in general only exact for linear functions on Cartesian meshes (see Lemma 3.4.4).

c) In general, the geometries of all considered test cases have corners and edges. Hence, the surface boundary of all geometries cannot be represented by a differentiable function.

d) Smoothness assumptions of solutions for the boundary value problems formulated in Section 2.4 are in general not possible.

These are reasons to conclude that with respect to mesh refinement studies the formal discretization order of two cannot be expected to be observed for solutions or values based on solutions. That is, with respect to the discrete solution and corresponding values such as lift and drag coefficients we only expect convergence of order one. More examples considering the accuracy towards the incompressible limit can be found in the article [46]. In this article in particular a turbulent flat plate case is presented, which exhibits quite exactly convergence of order one. It is assumed that this is due to the stagnation point singularity.

In the following Sections 6.9.1 – 6.9.4 we give examples for the extension of the discretization to the incompressible limit presented in Section 3.3. We start with a basic inviscid flow case to investigate the behavior for inflow Mach numbers down to $Ma_\infty = 10^{-4}$. Then, we show that the extension of the discretization to the incompressible limit is applicable to 3D transonic flow cases. As example we re-use the test case of Section 6.6. And then, in Sections 6.9.3 and 6.9.4 we investigate the technique for 3D high-lift flows. In particular the NASA Trap wing is discussed in detail.

6.9.1 Example 7: Inviscid flow over NACA 0012 airfoil

We consider a inviscid flow over the NACA 00012 airfoil. We perform the computations on a sequence of C-type structured meshes of dimension $160 \times 32, 320 \times 64$ and 640×128. The computed drag and lift coefficients are given in Table 6.9. As expected the non-modified scheme shows for all meshes a significant loss in accuracy starting at about a Mach number of $Ma = 0.01$. Lower inflow Mach numbers increase this loss in accuracy. Both the modification of Turkel (see 3.51) and the one of Rossow and Swanson (see Section 3.3.4) fix this problem, and consistent drag and lift coefficients are obtained also for the considered inflow Mach numbers below $Ma = 0.01$.

Figures 6.20, 6.21 and 6.22 show the distributions for the inflow Mach number 10^{-4}. Though, as expected, the quality of results improve with higher mesh densities for the non-modified scheme, even on the finest mesh of size 640×128 the results are not acceptable and comparable to the modified schemes. The non-modified scheme shows in particular a poor behavior on the trailing edge. With respect to C_p–distribution the modified schemes show comparable results. A closer look at the total pressure loss shows advantageous behavior of the Rossow/Swanson scheme.

Figures 6.23, 6.24 and 6.25 show the results for Mach number 10^{-3}, Figures 6.26, 6.27 and 6.28 for Mach number 10^{-2} and Figures 6.29, 6.30 and 6.31 for Mach number 10^{-1}. It is of no surprise that with increasing the inflow Mach number the results of the non-modified scheme improve. A closer look at the total pressure loss for $Ma = 10^{-1}$ even shows that the non-modified scheme exhibits better results compared to the Turkel modification. This is confirmed by the result for $Ma = 0.3$ given in Figures 6.32, 6.33 and 6.34. However, the Rossow/Swanson scheme also shows for $Ma = 0.3$ comparable results to the non-modified scheme for all mesh densities.

Having evaluated the accuracy with respect to expectation, we now use the extrapolation method explained in Section 6.9. Using (6.4), we compute the asymptotic values for the inflow Mach numbers $Ma_\infty = 0.3, Ma_\infty = 10^{-2}$ and $Ma_\infty = 10^{-4}$. The asymptotic values are given in Table 6.10. These values, in particular those for ω, do not show a clear behavior. There is a discrepancy between the lift and the drag coefficient. As expected, none of the results show the design order of the scheme. The results for the non-modified scheme and $Ma = 10^{-4}$ are so poor, that the system of equations (6.4) could not be solved. The modification of Turkel shows for the lift coefficient at all Mach numbers a convergence order of about 1.1 and for the drag coefficient of about 1.85. The scheme of Rossow/Swanson shows for the lift coefficient a convergence order of about 0.7 and for the drag coefficient of about 1.9. Nevertheless, the variety in the results shows that the given sequence of meshes was probably not suited to determine asymptotic values; that is, the assumptions for the estimate (6.6) are possibly not satisfied.

Mesh size	Ma_∞	Drag coefficient			Lift coefficient		
		Turkel	Ro.-Sw.	No mod.	Turkel	Ro.-Sw.	No mod.
160×32	0.3	$6.012 \cdot 10^{-4}$	$5.930 \cdot 10^{-4}$	$7.262 \cdot 10^{-4}$	0.24481	0.24745	0.25022
	0.1	$5.921 \cdot 10^{-4}$	$5.882 \cdot 10^{-4}$	$1.424 \cdot 10^{-3}$	0.23374	0.23632	0.24034
	0.01	$5.915 \cdot 10^{-4}$	$5.884 \cdot 10^{-4}$	$9.337 \cdot 10^{-3}$	0.23250	0.23511	0.23460
	0.001	$5.915 \cdot 10^{-4}$	$5.884 \cdot 10^{-4}$	$4.920 \cdot 10^{-2}$	0.23250	0.23511	0.17185
	0.0001	$5.915 \cdot 10^{-4}$	$5.884 \cdot 10^{-4}$	$3.519 \cdot 10^{-1}$	0.23249	0.23511	0.11856
320×64		Turkel	Ro.-Sw.	No mod.	Turkel	Ro.-Sw.	No mod.
	0.3	$3.476 \cdot 10^{-4}$	$3.195 \cdot 10^{-4}$	$3.071 \cdot 10^{-4}$	0.24631	0.24792	0.25061
	0.1	$3.307 \cdot 10^{-4}$	$3.035 \cdot 10^{-4}$	$4.819 \cdot 10^{-4}$	0.23510	0.23671	0.24097
	0.01	$3.290 \cdot 10^{-4}$	$3.011 \cdot 10^{-4}$	$3.146 \cdot 10^{-3}$	0.23384	0.23549	0.23990
	0.001	$3.290 \cdot 10^{-4}$	$3.010 \cdot 10^{-4}$	$1.391 \cdot 10^{-2}$	0.23383	0.23548	0.20077
	0.0001	$3.290 \cdot 10^{-4}$	$3.010 \cdot 10^{-4}$	$7.996 \cdot 10^{-2}$	0.23382	0.23548	0.07954
640×128		Turkel	Ro.-Sw.	No mod.	Turkel	Ro.-Sw.	No mod.
	0.3	$2.781 \cdot 10^{-4}$	$2.476 \cdot 10^{-4}$	$1.913 \cdot 10^{-4}$	0.24697	0.24819	0.25085
	0.1	$2.580 \cdot 10^{-4}$	$2.274 \cdot 10^{-4}$	$2.174 \cdot 10^{-4}$	0.23572	0.23696	0.24119
	0.01	$2.559 \cdot 10^{-4}$	$2.243 \cdot 10^{-4}$	$1.250 \cdot 10^{-3}$	0.23445	0.23572	0.24101
	0.001	$2.559 \cdot 10^{-4}$	$2.241 \cdot 10^{-4}$	$6.448 \cdot 10^{-3}$	0.23444	0.23571	0.20084
	0.0001	$2.558 \cdot 10^{-4}$	$2.241 \cdot 10^{-4}$	$3.361 \cdot 10^{-2}$	0.23444	0.23571	0.05032

Table 6.9: NACA 0012: Force coefficients for different Mach numbers

	$Ma_\infty = 0.3$		$Ma_\infty = 0.01$		$Ma_\infty = 0.0001$	
	drag coeff.	lift coeff.	drag coeff.	lift coeff.	drag coeff.	lift coeff.
	Turkel					
c_∞	$2.5186 \cdot 10^{-4}$	0.24749	$2.2769 \cdot 10^{-4}$	0.23496	$2.2749 \cdot 10^{-4}$	0.23498
κ	1.0156	0.42129	0.95834	0.31373	0.9508	0.27451
ω	1.8675	1.18442	1.8444	1.13535	1.8424	1.10109
	Rossow / Swanson					
c_∞	$2.2196 \cdot 10^{-4}$	0.248554	$1.9628 \cdot 10^{-4}$	0.236073	$1.9601 \cdot 10^{-4}$	0.236088
κ	1.3938	0.033598	1.3289	0.021228	1.3220	0.018296
ω	1.9275	0.799701	1.9034	0.724366	1.9020	0.685891
	No modification					
c_∞	$1.4709 \cdot 10^{-4}$	0.251234	$4.1302 \cdot 10^{-4}$	0.24130	no sol.	0.036803
κ	1.6008	0.020188	13.086	102.17439	no sol.	0.923016
ω	1.8557	0.70044	1.7072	2.25543	no sol.	0.417258

Table 6.10: Asymptotic force coefficients for NACA 0012 airfoil

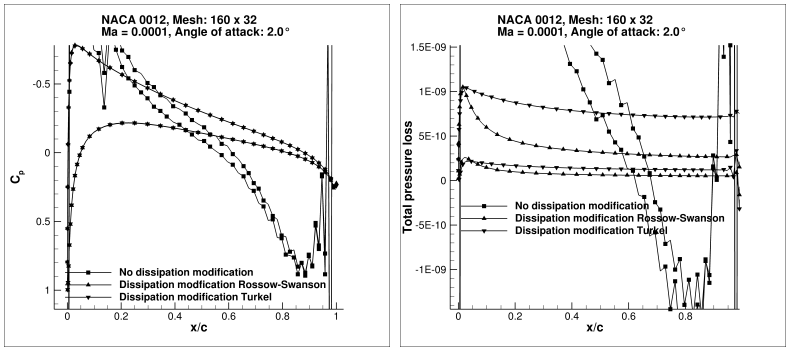

Figure 6.20: Mesh: 160×32, $Ma_\infty = 0.0001$: Computed C_p (left) and total pressure loss (right)

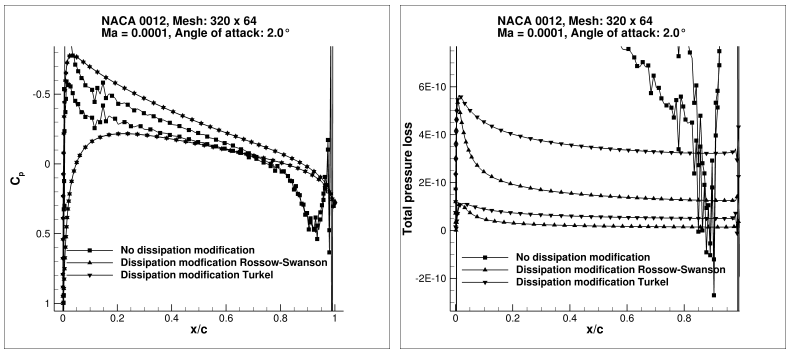

Figure 6.21: Mesh: 320×64, $Ma_\infty = 0.0001$: Computed C_p (left) and total pressure loss (right)

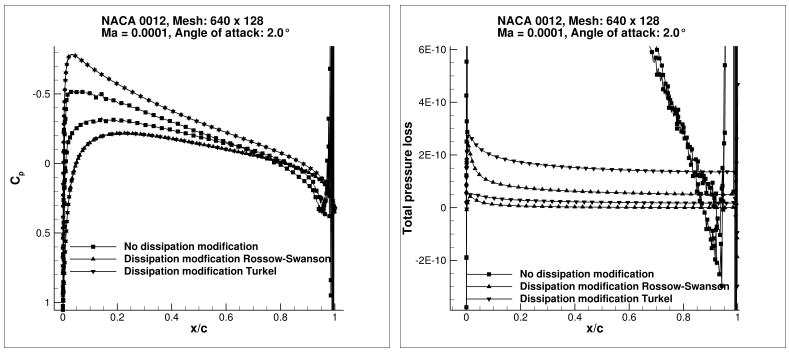

Figure 6.22: Mesh: 640×128, $Ma_\infty = 0.0001$: Computed C_p (left) and total pressure loss (right)

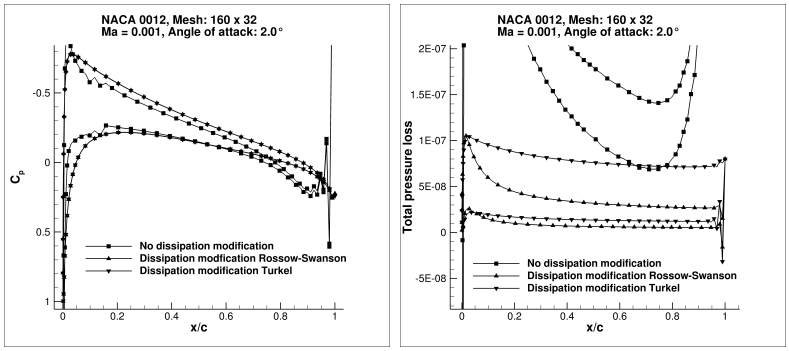

Figure 6.23: Mesh: 160×32, $Ma_\infty = 0.001$: Computed C_p (left) and total pressure loss (right)

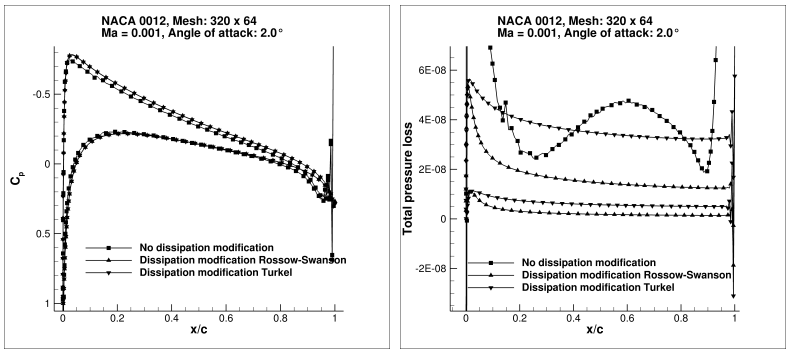

Figure 6.24: Mesh: 320×64, $Ma_\infty = 0.001$: Computed C_p (left) and total pressure loss (right)

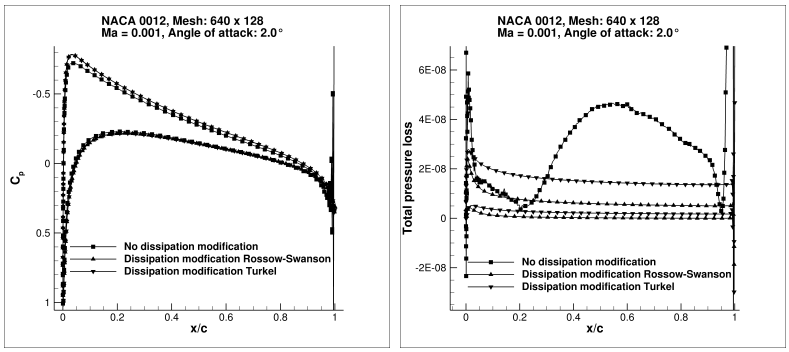

Figure 6.25: Mesh: 640×128, $Ma_\infty = 0.001$: Computed C_p (left) and total pressure loss (right)

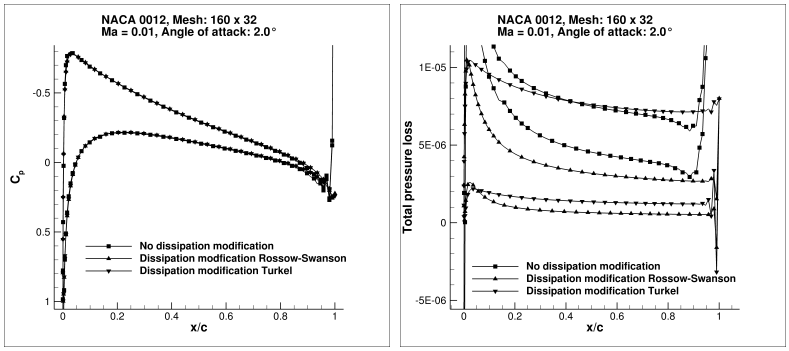

Figure 6.26: Mesh: 160×32, $Ma_\infty = 0.01$: Computed C_p (left) and total pressure loss (right)

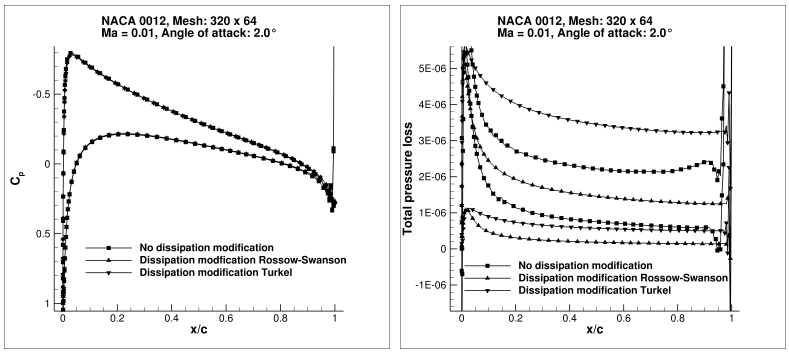

Figure 6.27: Mesh: 320×64, $Ma_\infty = 0.01$: Computed C_p (left) and total pressure loss (right)

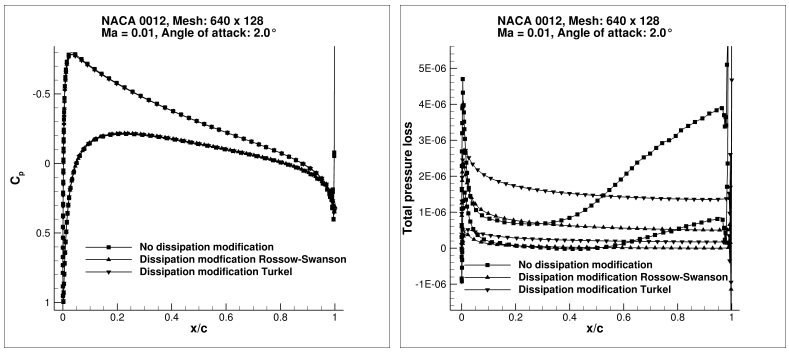

Figure 6.28: Mesh: 640×128, $Ma_\infty = 0.01$: Computed C_p (left) and total pressure loss (right)

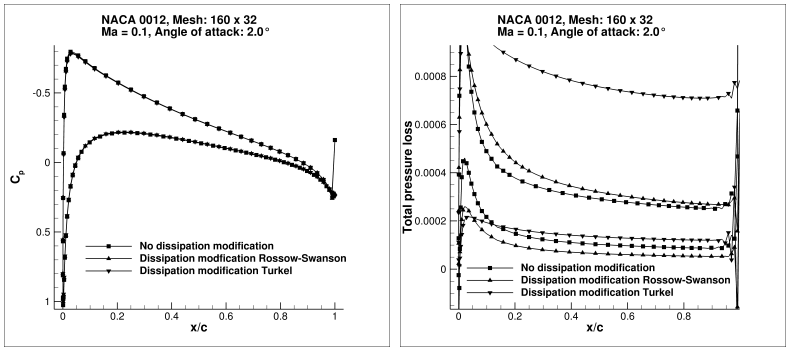

Figure 6.29: Mesh: 160×32, $Ma_\infty = 0.1$: Computed C_p (left) and total pressure loss (right)

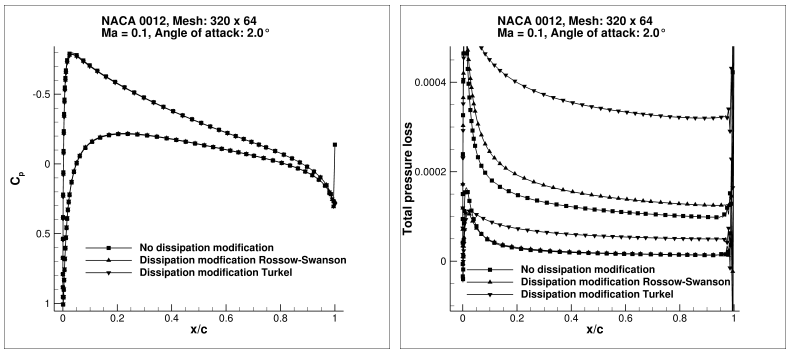

Figure 6.30: Mesh: 320×64, $Ma_\infty = 0.1$: Computed C_p (left) and total pressure loss (right)

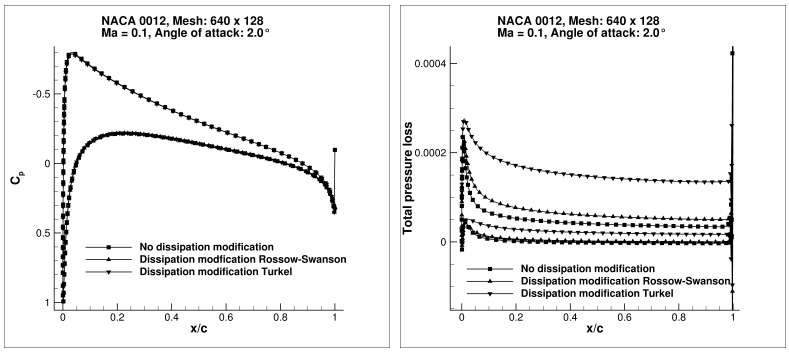

Figure 6.31: Mesh: 640×128, $Ma_\infty = 0.1$: Computed C_p (left) and total pressure loss (right)

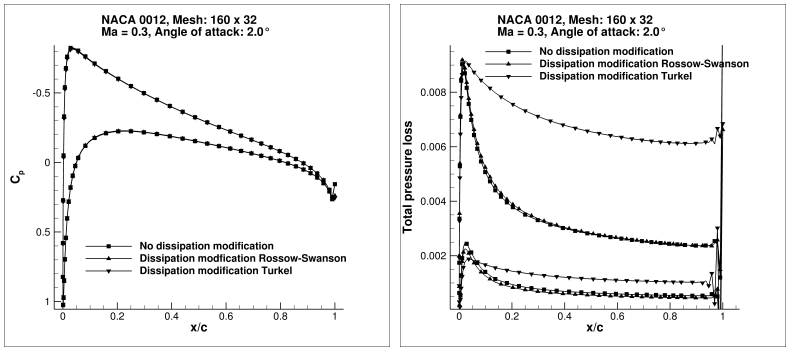

Figure 6.32: Mesh: 160×32, $Ma_\infty = 0.3$: Computed C_p (left) and total pressure loss (right)

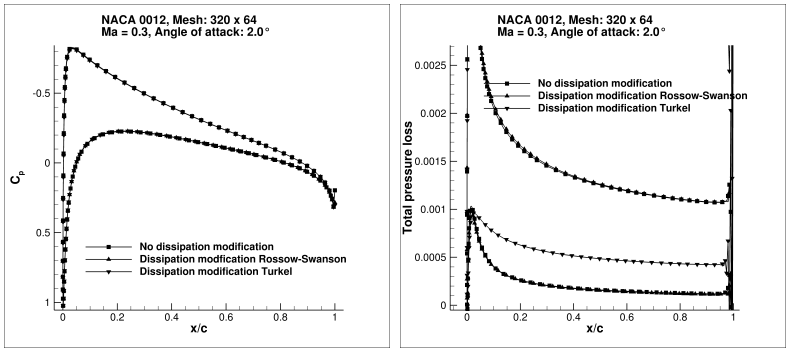

Figure 6.33: Mesh: 320×64, $Ma_\infty = 0.3$: Computed C_p (left) and total pressure loss (right)

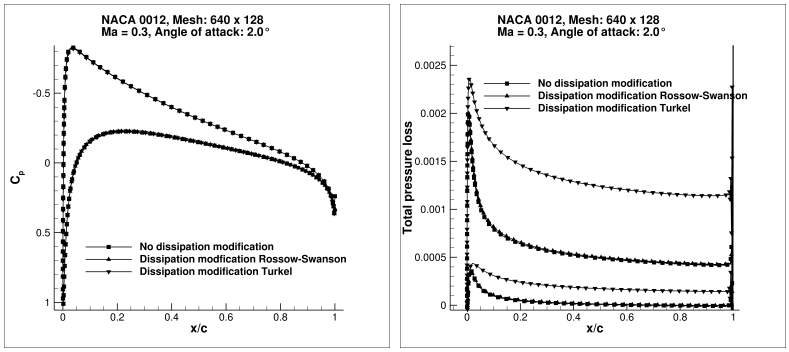

Figure 6.34: Mesh: 640×128, $Ma_\infty = 0.3$: Computed C_p (left) and total pressure loss (right)

6.9.2 Example 8: Transonic turbulent flow over a Common Research Model

To show the applicability of the different discretization schemes for transonic flows we reconsider the test case of Section 6.6. Instead of considering the target lift, for this investigation we consider a fixed angle of attack of 2.2°. For all flow cases a residual reduction of 14 orders of magnitude was obtained. All three considered discretization techniques show no problem with respect to convergence. A plot of the finest grid level used for the investigations with respect to the low Mach modifications and the convergence histories are shown in Figure 6.35.

The lift and drag coefficients for the schemes are given in Table 6.11. Application and solution of (6.4) gave the asymptotic values in Table 6.12. For this transonic test case larger discrepancies in results can only be observed for grid Level L1. The other grid levels as well as the extrapolated asymptotic values are in good agreement.

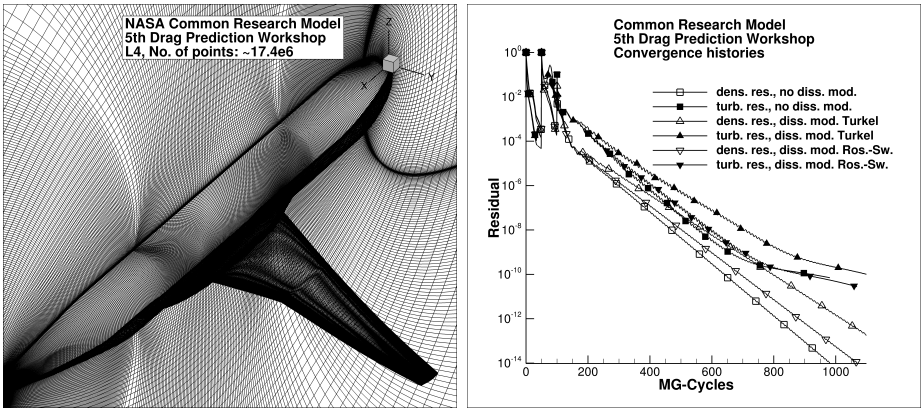

Figure 6.35: Left: Plot of hexahedral L4 mesh of NASA CRM; right: Convergence histories for hexahedral L4 mesh

6.9.3 Example 9: 3D high lift wing-body configuration

Since we are interested in flows which exhibit both incompressible and compressible regions, we now reconsider the 3D high-lift flow from Section 6.8. It is our goal to show that the low Mach modification of Rossow/Swanson is applicable to these kind of flows. To the author's experience, it is often these flow cases which are of interest in practice but where the traditional low Mach modification fails. A plot of the surface mesh can be found in Figure 6.36 (left).

The convergence history presented in Figure 6.36 (right) shows that using the low Mach modification of Rossow/Swanson causes no convergence problems, and robustness of the scheme as well as the observed convergence rate are comparable

Mesh	drag coefficient (no modification)	drag coefficient (mod. Turkel)	drag coefficient (mod. Rossow/Swanson)
L1	$2.442831 \cdot 10^{-2}$	$2.425625 \cdot 10^{-2}$	$2.484298 \cdot 10^{-2}$
L2	$2.499070 \cdot 10^{-2}$	$2.495923 \cdot 10^{-2}$	$2.506497 \cdot 10^{-2}$
L3	$2.517880 \cdot 10^{-2}$	$2.516215 \cdot 10^{-2}$	$2.521016 \cdot 10^{-2}$
L4	$2.534337 \cdot 10^{-2}$	$2.533301 \cdot 10^{-2}$	$2.535941 \cdot 10^{-2}$
	lift coefficient (no modification)	lift coefficient (mod. Turkel)	lift coefficient (mod. Rossow/Swanson)
L1	$4.844779 \cdot 10^{-1}$	$4.792998 \cdot 10^{-1}$	$4.911367 \cdot 10^{-1}$
L2	$5.025033 \cdot 10^{-1}$	$5.020096 \cdot 10^{-1}$	$5.026062 \cdot 10^{-1}$
L3	$5.070862 \cdot 10^{-1}$	$5.068201 \cdot 10^{-1}$	$5.071321 \cdot 10^{-1}$
L4	$5.106439 \cdot 10^{-1}$	$5.104293 \cdot 10^{-1}$	$5.108746 \cdot 10^{-1}$

Table 6.11: Computed drag and lift coefficient of DPW5 Common Research model

	No modification		Turkel		Rossow / Swanson	
	drag coeff.	lift coeff.	drag coeff.	lift coeff.	drag coeff.	lift coeff.
c_∞	$2.5558 \cdot 10^{-2}$	$5.1407 \cdot 10^{-1}$	$2.5534 \cdot 10^{-2}$	$5.1364 \cdot 10^{-1}$	$2.5687 \cdot 10^{-2}$	$5.1508 \cdot 10^{-1}$
κ	0.537838	62.55723	0.9511	103.60422	0.057378	26.85162
ω	1.407716	1.7656	1.522156	1.86811	0.929182	1.57627

Table 6.12: Asymptotic force coefficients for NASA CRM of 5th Drag Prediction Workshop

to the non-modified scheme. The residuals of the scheme of Turkel stalled after about 6 orders of magnitude. Also, an adaptation of the CFL number or several other parameters of the solution scheme were not able to prohibit this failure of convergence. Moreover, the convergence plots of lift and drag coefficient shown in Figure 6.37 exhibit different values. This is clearly an indicator that for such a complex flow the discrete solution obtained is not in the asymptotic range. The difference in the C_p and C_f distributions given in Figure 6.38 confirms this indication. Nevertheless, this example demonstrates the improved robustness of the Rossow/Swanson scheme compared to the original Turkel scheme.

Referring to the conclusion made in Section 6.8, we emphasize again the necessity for for fully converged solutions. Hence, the non converged solution of the Turkel scheme needs to be handled with care and the improved robustness of the Rossow/Swanson scheme clearly has its advantages.

6.9.4 Example 10: NASA TRAP Wing

As a final example to investigate the extension of the discretization scheme to the incompressible limit, we examine the NASA Trap Wing considered at the first

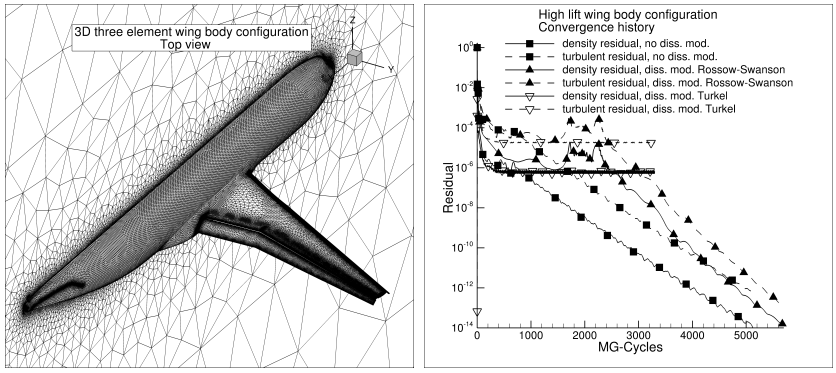

Figure 6.36: 3D high lift wing-body configuration: View on the mesh (left), convergence history (right)

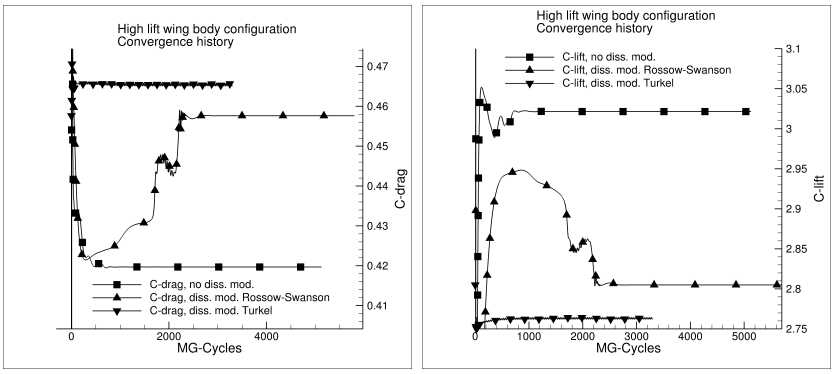

Figure 6.37: 3D high lift wing-body configuration: Convergence history of drag coefficient (left) and of lift coefficient (right)

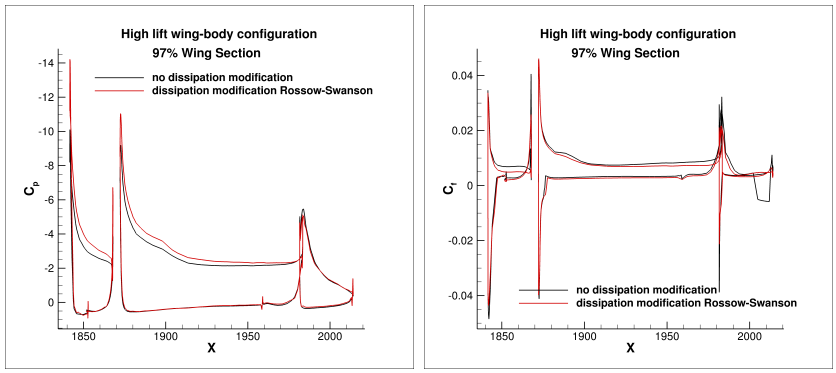

Figure 6.38: 3D high lift wing-body configuration: C_p–distribution (left) and C_f–distribution at the 97% span location with non-modified and modified dissipation

AIAA High-Lift prediction workshop [83]. For the numerical computations, we used a sequence of three meshes. The meshes were generated using VGrid and are marked as UH6 in Table 2 of [83]. Rough data of these meshes is given in Table 6.13. A surface plot of the finest mesh used is given in Figure 6.40 (right). The physical

Mesh	No. of points	No. of elements
Coarse (C)	3727008	10169092
Medium (M)	11047965	38017477
Fine (F)	32445391	127443165

Table 6.13: NASA TRAP Wing: Mesh data

parameters of the test case are as follows:

- Geometry: NASA TRAP Wing
- Reynolds number: $Re = 4.3 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.2$

Computations were done for the angles of attack 13°, 28°, 32°, 34°and 37°. Note that for all numerical experiments a residual reduction of 14 orders of magnitude was possible, even for the fully separated flow at 37°angle of attack. None of the computed solutions showed dependency on the initial conditions, and a standard procedure using full multigrid for the start-up phase was successful in all computations. Also, though tried for a given discretization on a given mesh, multiple solutions were not obtained as reported by Kamenetskiy et al [33]. Figure 6.39 shows some of the convergence histories only on the finest mesh considered, in particular for the highest angle of attacks 34°and 37°as well as for the 28°case. For the coarser meshes convergence of the computations was in general straightforward, and on all meshes a CFL number between 100 and 1000 could be reached.
The determined force coefficients of lift and drag are given in Table 6.14. A plot of the $C_L(\alpha)$ branch is given in Figure 6.41, and comparisons with measurements are done. It can be directly observed that for the given grids, that is a given resolution, the behavior of the non-modified scheme and the modified scheme for high angles of attack is totally different. The non-modified discretization predicts a maximum lift between 34°and 37°, whereas the modified scheme exhibits the maximum lift between 32°and 34°. In particular, the medium mesh shows for the modified scheme an unexpected behavior, which does not correspond to the results for the coarse and the fine mesh. The differences for all other angles of attack for both discretizations are minor.
C_p distributions are shown at the 50% and 98% wing section and compared with experimental data. For the angle of attack 13°these distributions are given in Figures 6.42, 6.43 and 6.44, for the angle of attack 28°in Figures 6.45, 6.46 and 6.47,

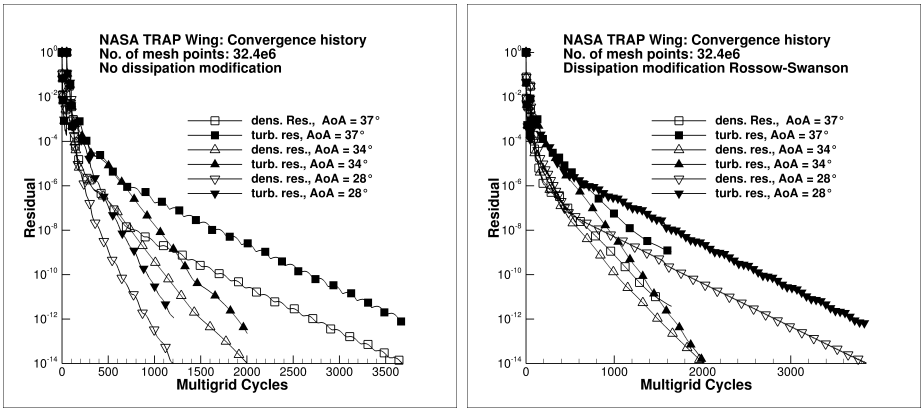

Figure 6.39: Convergence histories for NASA TRAP Wing

		no diss. mod.		diss. mod. Ros.-Swan.	
AoA	Mesh	drag coeff.	lift coeff.	drag coeff.	lift coeff.
13°	C	$3.285988 \cdot 10^{-1}$	1.973706	$3.253006 \cdot 10^{-1}$	1.989436
	M	$3.270158 \cdot 10^{-1}$	2.010587	$3.275418 \cdot 10^{-1}$	2.022306
	F	$3.288310 \cdot 10^{-1}$	2.028653	$3.275920 \cdot 10^{-1}$	2.027132
28°	C	$6.689967 \cdot 10^{-1}$	2.812827	$6.663658 \cdot 10^{-1}$	2.823602
	M	$6.819167 \cdot 10^{-1}$	2.899304	$6.843836 \cdot 10^{-1}$	2.909559
	F	$6.892306 \cdot 10^{-1}$	2.931708	$6.871403 \cdot 10^{-1}$	2.928343
32°	C	$7.598812 \cdot 10^{-1}$	2.939004	$7.498742 \cdot 10^{-1}$	2.876154
	M	$7.770928 \cdot 10^{-1}$	3.036963	$7.305732 \cdot 10^{-1}$	2.488504
	F	$7.791266 \cdot 10^{-1}$	3.064850	$7.720406 \cdot 10^{-1}$	3.036663
34°	C	$8.028974 \cdot 10^{-1}$	2.933732	$7.564492 \cdot 10^{-1}$	2.234626
	M	$8.124077 \cdot 10^{-1}$	3.046266	$6.588863 \cdot 10^{-1}$	1.644493
	F	$8.146982 \cdot 10^{-1}$	3.091051	$7.689900 \cdot 10^{-1}$	2.196198
37°	C	$8.387106 \cdot 10^{-1}$	2.078015	$8.042444 \cdot 10^{-1}$	1.871872
	M	$8.260612 \cdot 10^{-1}$	1.943235	$7.927863 \cdot 10^{-1}$	1.657968
	F	$8.351719 \cdot 10^{-1}$	1.895830	$8.251337 \cdot 10^{-1}$	1.844026

Table 6.14: Computed drag and lift coefficients for NASA Trap Wing

for the angle of attack 34°in Figures 6.48, 6.49 and 6.50, and for the angle of attack 37°in Figures 6.51, 6.52 and 6.53.

For the 13°and 28°cases the C_p-distribution at the 50% looks similar for all mesh sizes. For the 98% wing section an assertion is not straightforward, and the results are difficult to compare. Obviously, the differences on the main wing and the flap are larger when compared with the slat. Due to the offset to the experimental data, an assertion with respect to accuracy cannot be done. In particular, on the finest

mesh considered there is again good agreement for the Rossow/Swanson modified and the non-modified schemes.

The situation changes dramatically when considering the 34°case. Already observed in the lift coefficient (see Figure 6.41), the non-modified scheme predicts a significantly different lift than the Rossow/Swanson modified scheme. This goes along with the observation that for all meshes the computed C_p distribution looks different on the slat, the main wing and on the flap. Moreover, the non-modified scheme represents significantly better the experimental data than the Rossow/Swanson modified scheme. The differences in the computed flow fields are further illustrated in Figure 6.54. In the left figure the streamlines of the velocity field for the non-modified scheme are given. The prediction of the flow field is smooth without large separations. This situation is totally different when compared with the flow field computed with the Rossow/Swanson modified scheme. Here, the velocity streamlines predict a huge separation over the wing. Again, note that both figures represent fully converged solutions on the given grid. They are different with respect to the discretization of the artificial viscosity terms.

For the 37°cases, in particular on the finest mesh, both schemes predict about the same C_p-distribution. Also, as shown in Figure 6.55, the streamlines of the flow for the non-modified scheme and the modified one are in good agreement, and both discrete solutions predict a large flow separation over the wing. Though the flow is fully separated, the obtained steady-state solution still shows quite good agreement with the experimental data. Due to the complexity of the flow and since it can be assumed to be fully separated and unsteady, such an agreement might be accidental.

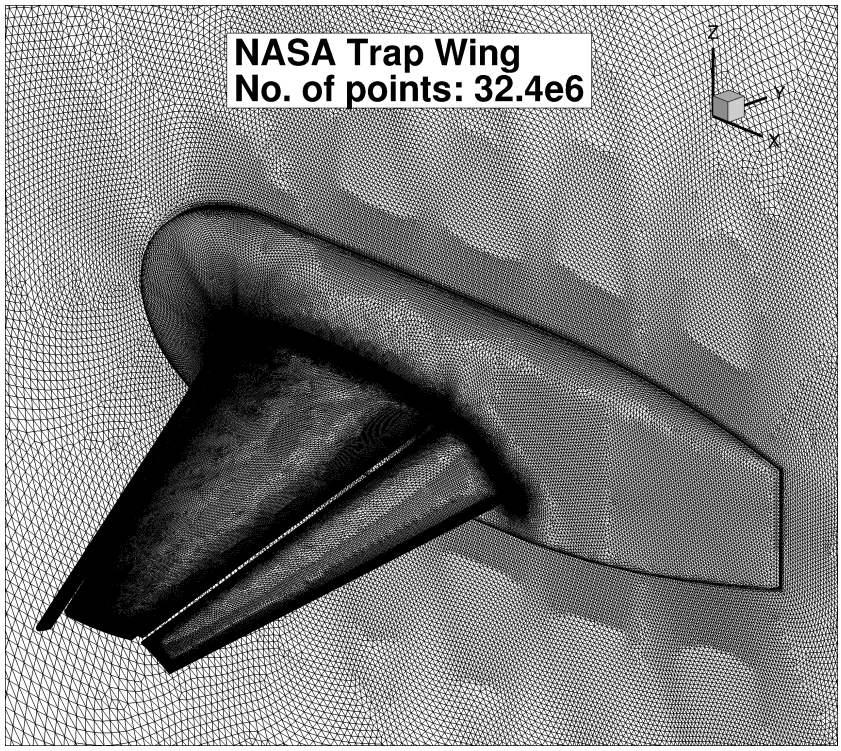

Figure 6.40: NASA TRAP Wing: View on the mesh

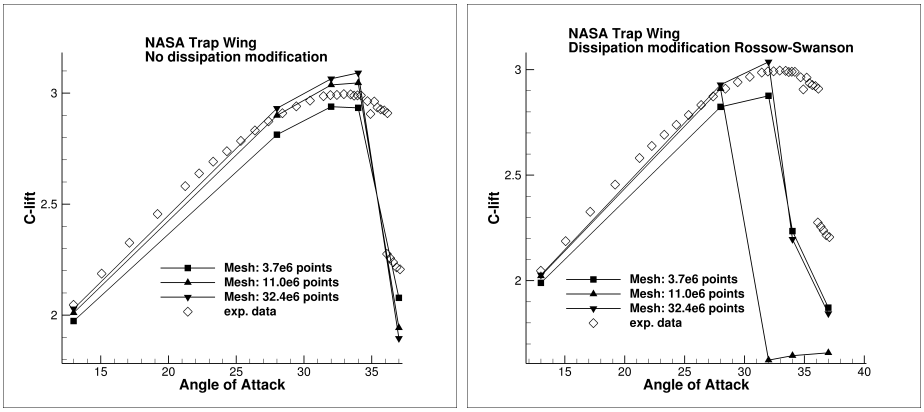

Figure 6.41: NASA TRAP Wing: Representation of $C_L(\alpha)$ branch without dissipation modification (left) and with dissipation modification(right)

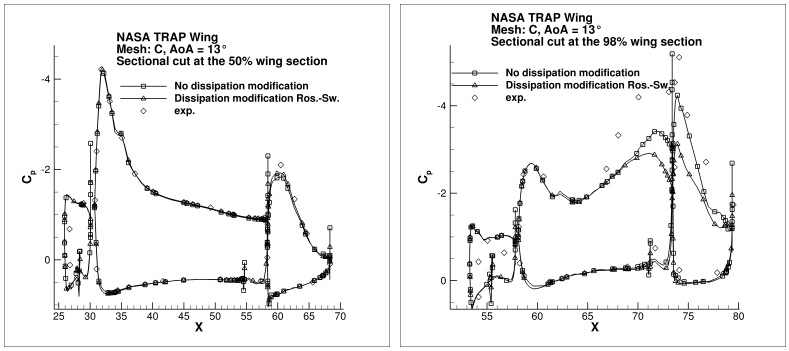

Figure 6.42: NASA TRAP Wing, coarse mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 13°

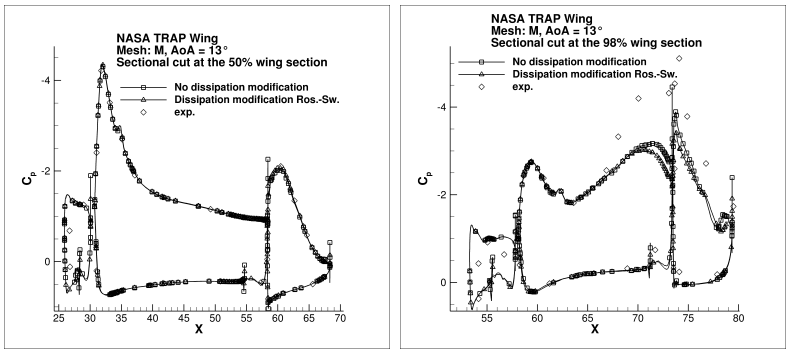

Figure 6.43: NASA TRAP Wing, medium mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 13°

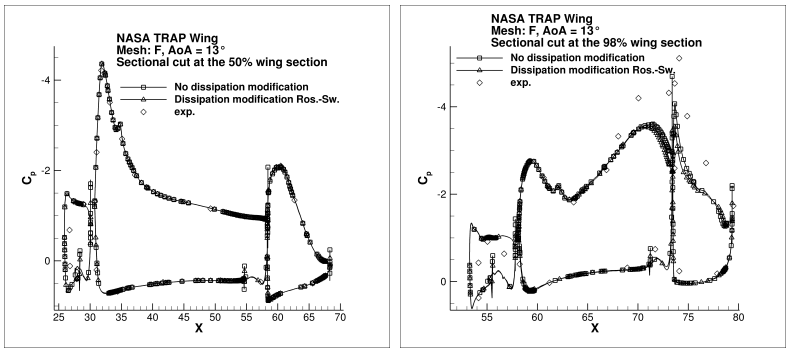

Figure 6.44: NASA TRAP Wing, fine mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 13°

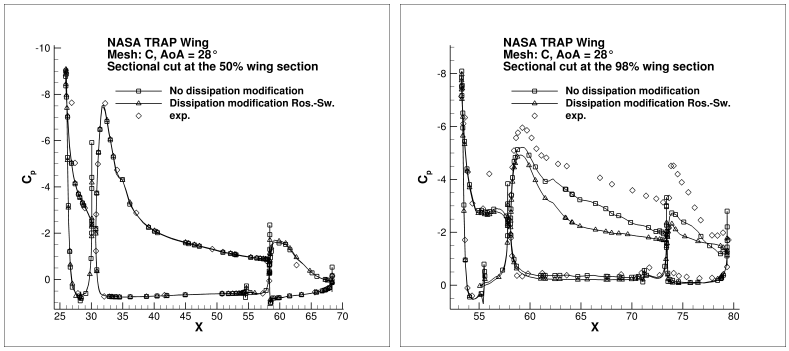

Figure 6.45: NASA TRAP Wing, coarse mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 28°

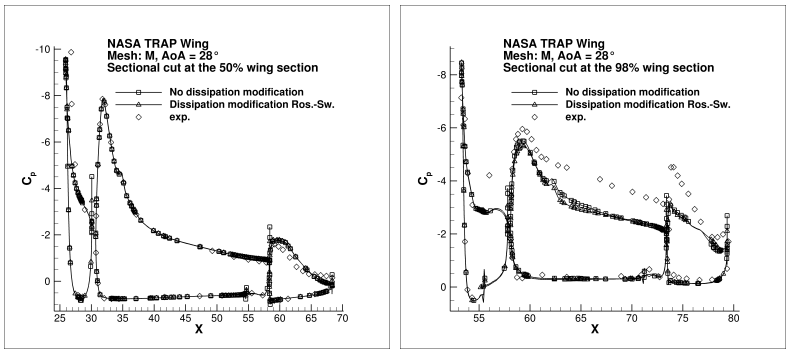

Figure 6.46: NASA TRAP Wing, medium mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 28°

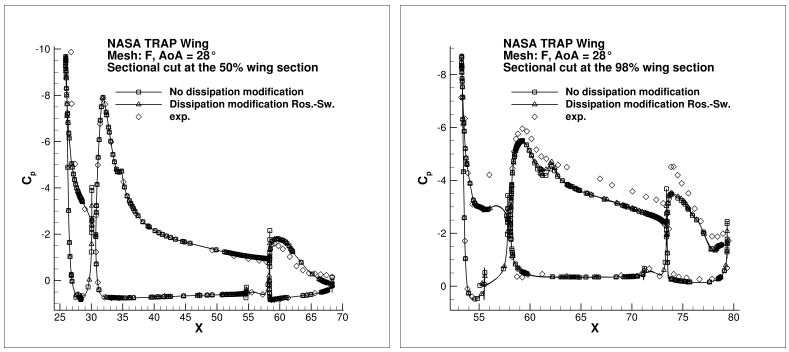

Figure 6.47: NASA TRAP Wing, fine mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 28°

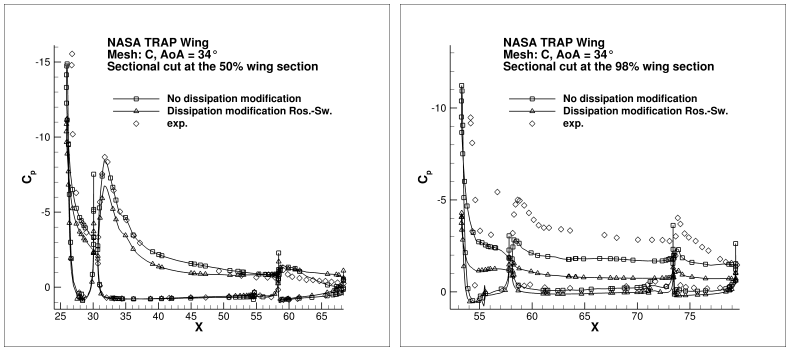

Figure 6.48: NASA TRAP Wing, coarse mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 34°

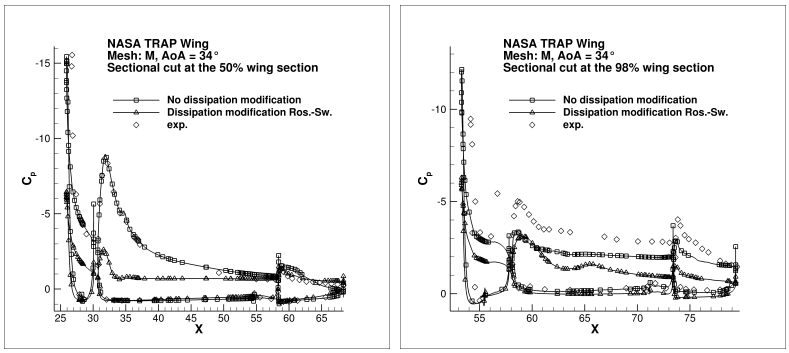

Figure 6.49: NASA TRAP Wing, medium mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 34°

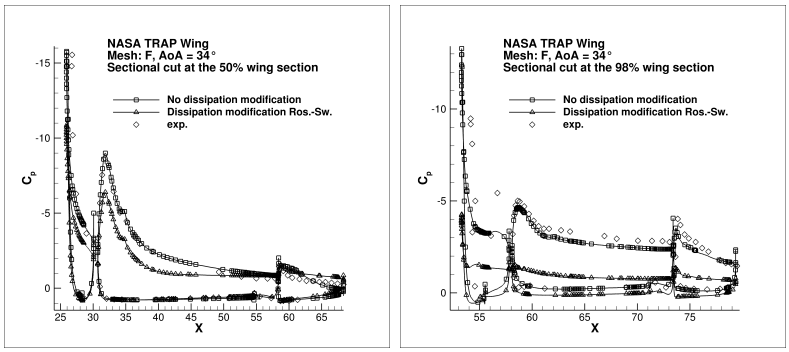

Figure 6.50: NASA TRAP Wing, fine mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 34°

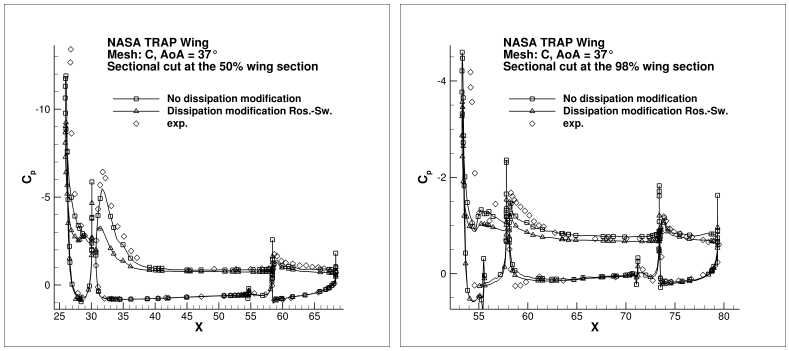

Figure 6.51: NASA TRAP Wing, coarse mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 37°

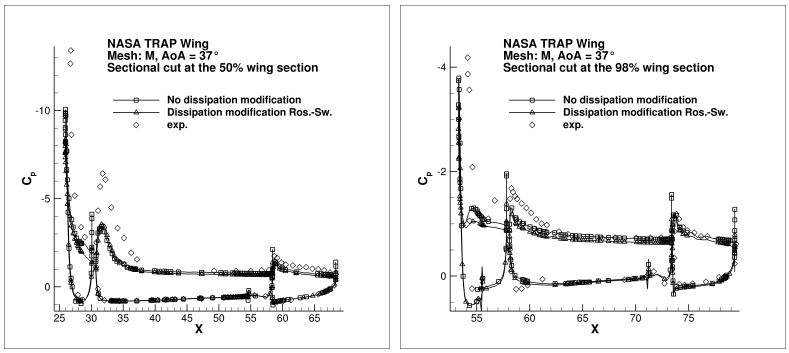

Figure 6.52: NASA TRAP Wing, medium mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 37°

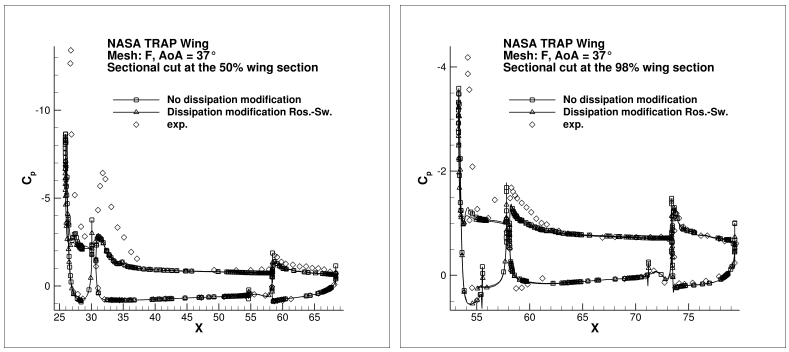

Figure 6.53: NASA TRAP Wing, fine mesh: C_p–distribution at the 50%(left) and 98%(right) wing position and angle of attack 37°

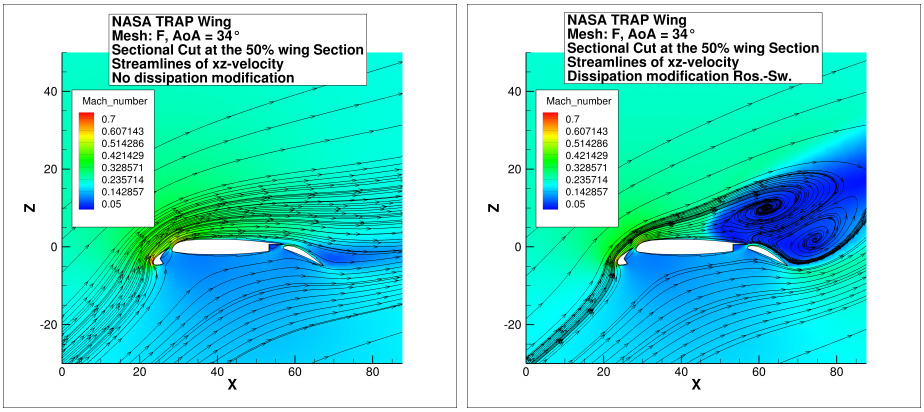

Figure 6.54: NASA TRAP Wing: Streamlines of xz-velocity at the 50% wing position and angle of attack 34°

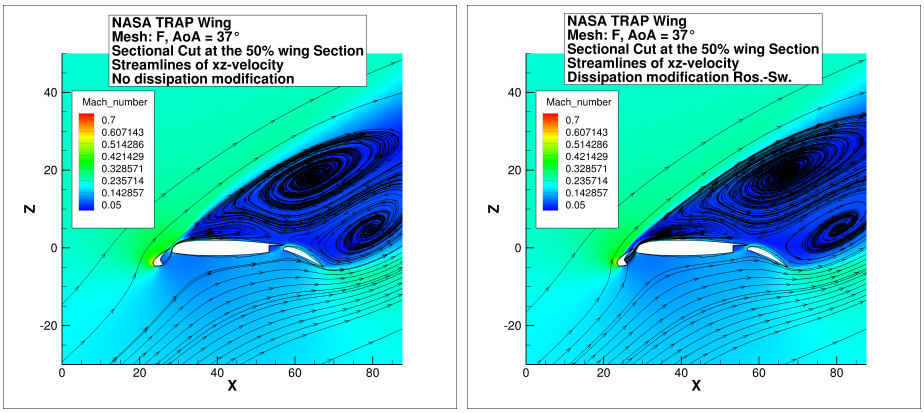

Figure 6.55: NASA TRAP Wing: Streamlines of xz-velocity at the 50% wing position and angle of attack 37°

6.10 Example 11: Laminar flow over NACA0012 airfoil

So far, for all examples shown we have not yet exploited the full algorithmical potential suggested in Figure 5.1. This example is dedicated to this topic. It corresponds to the exterior viscous flow problem formulated in Section 2.4. The relevant physical conditions are

- Geometry: NACA 0012 airfoil
- Reynolds number: $Re = 5000$

- Inflow Mach number: $M_\infty = 0.5$
- Angle of attack: AoA $= 0.0\,^\circ$.

The mesh used is a C-type structured mesh of size 512×256. For a detailed analysis of this test case we refer to [90]. In this referred article this example is investigated for the additional angles of attack $1.0\,^\circ$, $2.0\,^\circ$and $3.0\,^\circ$up to a mesh size of 4096×2048. It is out of the scope of this thesis to repeat such results. But it has turned out, that for thee development of robust and reliable solution algorithms, these test cases are of major importance. On the one hand these test cases do not show additional complexity since turbulence is not modeled. On the other hand, due to the large separations at the trailing edge of the airfoil (see [90] for more details), in the context of this article only regularized Newton-kind smoothers such as Algorithm 3 were able to find a steady state solution for the finest mesh sizes considered. Smoothers like the LU-SGS method (see Theorem 5.3.4) were not even able to find a steady state, already on rather coarse meshes. We refer to Section 7.3.1 and [45] where analytical reasons for such behavior are discussed.

In this thesis, to approximate a steady state solution for the mesh of size 512×256, we apply the nonlinear multigrid Algorithm 5.1.6 with smoother (5.42). Instead of prescribing the number of steps to approximately solve (5.40), we choose as truncation criterion (5.39) in combination with

$$\begin{aligned} \varepsilon(k) &= 10^{-12}, \\ \max_{\text{iter}} &= 50. \end{aligned} \tag{6.7}$$

It is the goal of this example to demonstrate the potential of linear multigrid Algorithm 5.2.2 together with a residual based truncation criterion.

Therefore, we performed two iterations. In the first test we used as linear solver a single grid iteration, and in the second test we applied a $4V$ linear multigrid cycle. Figures 6.56 (left) and 6.56 (right) show the convergence histories. In Figure 6.56 (left) one observes that for the fixed truncation criterion (6.7) the convergence history of the outer nonlinear residual is (almost) independent of the inner linear solution method. This observation suggests that indeed a residual based truncation criterion is advantageous compared to a fixed number of iterations. Possible deficiencies of the inner linear solution method are compensated. To illustrate convergence of the inner single grid and linear multigrid iteration, Figure 6.56 (right) shows a representative section of the inner linear residuals. Obviously, as expected the single grid iteration is algorithmically much less efficient. The $4V$ multigrid cycle requires much less iterations until the truncation criterion of 10^{-12} is satisfied. For a fixed number of inner iterations, say for example 10, from Figure 6.56 (right) one can also conclude that the linear multigrid cycle reduces the linear residual by about $5 - 6$ orders of magnitude, whereas the single grid iteration only performs a

reduction of about 2 – 3 orders of magnitude. Such a large difference in the residuals may lead to a totally different update for the outer nonlinear loop and also a different behavior of the outer nonlinear multigrid Algorithm 5.1.6.

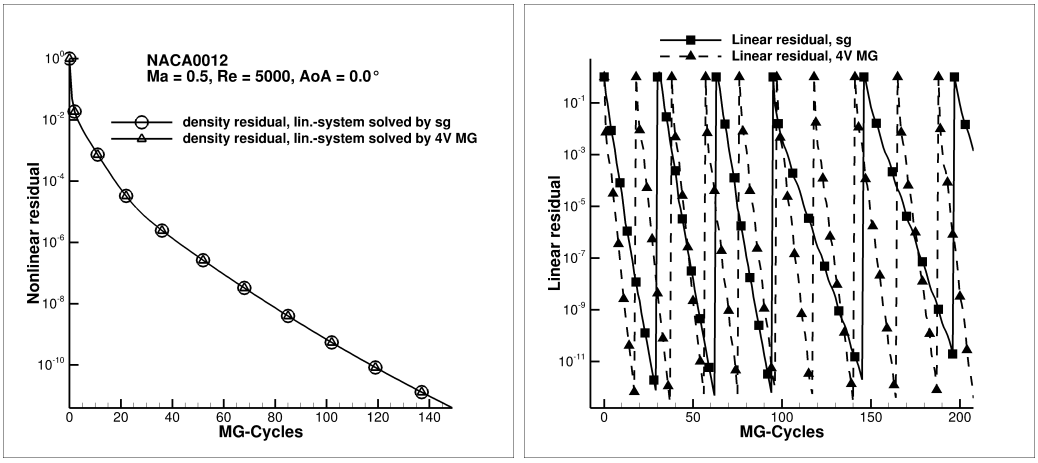

Figure 6.56: Left: Convergence history of nonlinear residual, right: Convergence history of linear residual

Though the inclusion of linear multigrid as well as a residual based truncation criteria show algorithmical improvements, it is important that within our implementation these additional features yield a significant increase in the total computational time. Hence, these techniques have not yet been applied to large scale 3D simulations. It is of major importance and future work to improve the efficiency of such algorithmical features.

6.11 Application of $k\omega$-model

This section, and the examples considered in Subsections 6.11.1, 6.11.2, 6.11.3, 6.11.4 and 6.11.5 deal with the application of implicit methods to the two equation $k\omega$-model (2.22) presented in Section 2.2.2. Within the community of computational fluid dynamics it seems to be accepted point of view, that it is in general possible to approximate solutions of the corresponding exterior turbulent flow problem formulated in Section 2.4 modeling turbulence by a $k\omega$-model.

Considering an unstructured finite volume code it was the experience in the author's implementation of this model, that it was not straightforward at all to construct a robust and reliable solution method for this turbulence model taking care of the following demands:

a) Implement the model free of direct limiters for the variables k and ω.

b) Avoid a reformulation of several parts of the model.

Note that often codes used in computational fluid dynamics have hard limitations for k and ω to ensure positivity of the variables. The necessity of positivity is a direct consequence of (2.54). In general, the implementations do not work without such limitations. Neglecting those limitations often as a sudden the computation ends with "Not a number". Let us shortly discuss the reasons for the demands formulated above. Using such limitations has several major drawbacks:

a) Iterations do not converge to steady state, since artificially values for k and ω are manipulated. Mathematically, the degree of freedom, where the limiter acts, is then a side condition and it would be required to remove it from the degrees of freedom.

b) Even, in case the iteration converges to a steady state, one finally has to check that the limiter is not active. Otherwise one has maybe considerably changed the outcome of the model.

c) The values the limitations are based on are often ad hoc values experienced by the behavior of certain examples. Hence, it can be assumed that there is no generality in their construction. Therefore these techniques may only work for a few similar examples.

d) One rarely finds information about the technical details of the limitation. Reproducibility of results is therefore often either hard or even impossible.

e) Limitation may change both the solution and the analytical formulation of the model. Hence, the introduction of these techniques may be also viewed as turbulence modeling.

For example, in the framework of discontinuous Galerkin codes [4, 28] the original $k\omega$-model has been reformulated in an analytical way. Hence, actually these authors have changed the original set of equations already on its analytic foundation. Unfortunately, the authors leave it open if these modifications are active for steady state solutions, and if they are, they leave it open in which way these limiters influence the prediction quality of the original model.

Though a turbulence model is only a model, and modifications might be allowed, one has to admit that finally one solves for a different set of equations and comparisons to the original formulation are not straightforward or even impossible. To avoid all such problems, within our framework the only limitation implemented is mentioned in (2.24). And such limitation is mentioned also by other authors, for example [60, 62], and Section 6.11.1 gives an example for the necessity of this limitation.

To deal with the problem of positivity of k and ω we simply introduced a damping of the updates. For example, Algorithm (5.42) gives for the variables k_i and ω_i, $i =$

$1, \ldots, N_{elem}$, the updates

$$k_i^{(j)} = k_i^{(0)} - \Delta k_i, \tag{6.8a}$$
$$\omega_i^{(j)} = \omega_i^{(0)} - \Delta \omega_i, \tag{6.8b}$$

where $(\Delta k_i, \Delta \omega_i)$ denotes the symbol for ith entry of vector one obtains evaluating $\alpha_{j+1,j}\mathbf{Prec}^{-1}\mathbf{R}(\mathbf{W}^j)$. The direct application of (6.8) often yield negative values in particular for k. Most often this was observed for the high-lift test cases. Therefore, we replaced (6.8) by

Algorithm 4 Update for $k\omega$-model

```
1: procedure LOOP OVER ALL MESH POINTS TO UPDATE k AND ω
2:     for i = 1, ..., N_elem do
3:         s_n = 1
4:         for n = 1, 2, ... do
5:             k_i^new = k_i^(0) − s_n Δk_i
6:             if k_i^new > 0 then
7:                 k_i^(j) = k_i^new
8:                 break
9:             else
10:                s_{n+1} = s_n / 2
11:    for i = 1, ..., N_elem do
12:        s_n = 1
13:        for n = 1, 2, ... do
14:            ω_i^new = ω_i^(0) − s_n Δω_i
15:            if ω_i^new > 0 then
16:                ω_i^(j) = ω_i^new
17:                break
18:            else
19:                s_{n+1} = s_n / 2
```

Naturally, Algorithm 4 represents some kind of damped Newton method introducing a further effect of regularization. The undesired side effect is, that the updates may become arbitrary small yielding an overall convergence corruption. However, for none of the considered test cases stall of convergence has been observed so far. Compared with many others methods tried to ensure positivity of k and ω Algorithm 4 was within the author's implementation always superior. The simplicity is a further argument for Algorithm 4. But application of Algorithm 4 cannot guarantee convergence. Hence, future work needs to focus on other mechanisms to ensure positivity of k and ω without reformulating the $k\omega$-model itself.

Nevertheless, compared to exterior turbulent flow problem in combination with the Spalart-Allmaras model it turned out that a combination with a $k\omega$-model yield

to much harder problems to set up a reliable and robust solution method. The implementation of the model together with the solution algorithm must be viewed as proprietary, and far away from being usable for large scale problems. It seems that the stiffness inherent in this set of equations together with the unresolved analytical background in which way to choose proper boundary values, the understanding for these equations is at a very primitive level. And, finally, since the error introduced by the turbulence model cannot be assessed, it is an interesting question if it is well worth the effort to try to develop suitable algorithms to approximately solve for this set of equations.

However, in the following examples it is our goal to demonstrate that in principal the solution methodology suggested in this thesis is at least for some basic flow problems possible to deal with this kind of equations. For all the examples considered in the following we apply Algorithm (5.42) in combination with (5.36).

6.11.1 Example 12: CASE 10, RAE 2822

We consider turbulent flow over the RAE 2822 airfoil. The relevant physical conditions are:

- Geometry: RAE 2822 airfoil
- Reynolds number: $Re = 6.2 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.75$
- Angle of attack: AoA = 2.81 °.

Turbulence is modeled using the $k\omega$-model (2.52). For the computation the mesh of size 320×64 from Section 6.3 is used.
To demonstrate the necessity of the production limiter (2.24), we visualize the eddy viscosity in the neighborhood of the airfoil. For all the results shown machine accurate results could be obtained on the given mesh. The convergence history is given in Figure 6.58. The results are given in Figure 6.57. Without production limiter the $k\omega$-model in $Pr_{k,(k,\omega)}$ shows an excess of eddy viscosity in a region where the shock interacts with the outer region of the boundary layer. This effect is eliminated by using in the term $Pr_{k,(k,\omega)}$ the production limiter (2.24). Hence, the production limiter (2.24) in the $k\omega$-model is not introduced for a possible better numerical behavior, but it has a severe impact on a corresponding solution and it is therefore an important part of the model itself, that is part of the analytical description of the model. And the flow prediction reacts sensitive with respect to this reformulation.

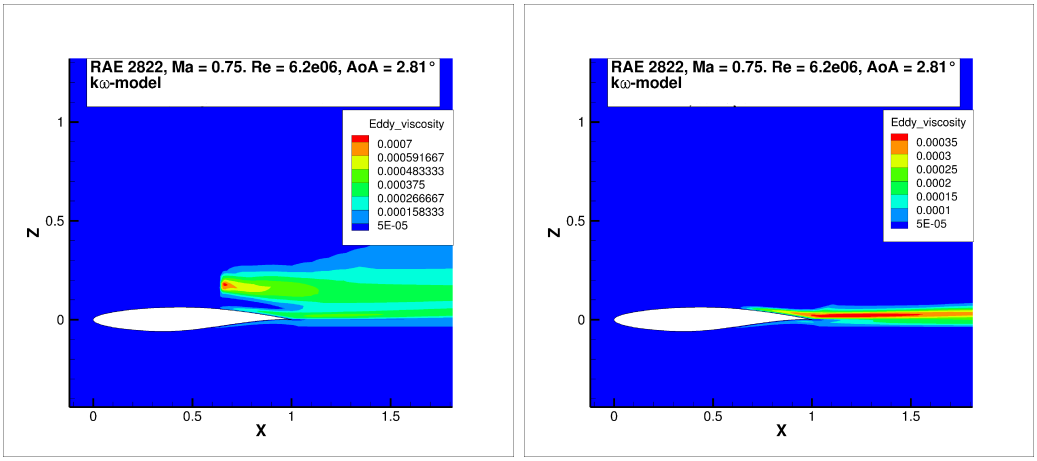

Figure 6.57: Eddy viscosity in the neighborhood of the airfoil without (left) and with (right) production limiter

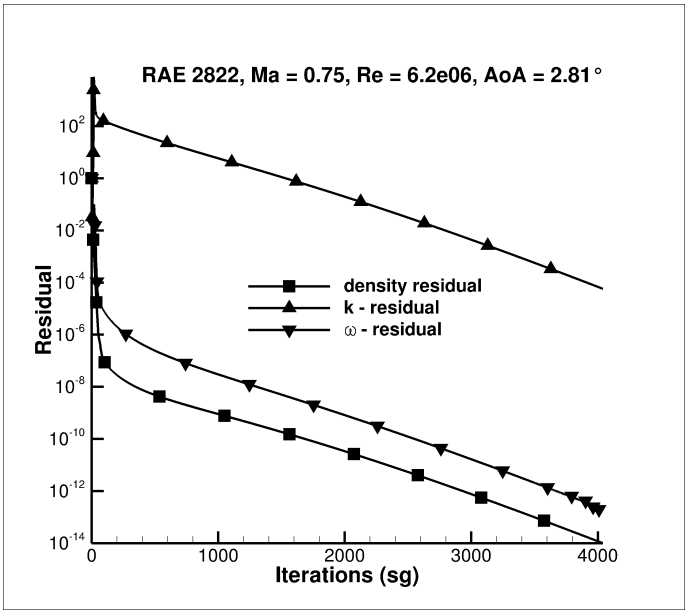

Figure 6.58: Convergence history

6.11.2 Example 13: CASE 9, RAE 2822

We reconsider the example of Section 6.3, now in combination with the $k\omega$-model. Again, the relevant physical conditions are:

- Geometry: RAE 2822 airfoil
- Reynolds number: $Re = 6.5 \cdot 10^6$

- Inflow Mach number: $M_\infty = 0.73$
- Angle of attack: AoA = 2.79 °.

Turbulence is modeled using the $k\omega$-model (2.52). We use again the mesh of size 320×64. This test case demonstrates the influence of the balance between the mean flow equations (2.1a) and the turbulent flow equations (2.22). Solving the nonlinear algebraic system of equations (3.97) weakly coupled, we show that it is in particular the turbulent flow equations, which rule the overall convergence rate. This serves as an indicator for the stiffness introduced by this equation into the full set of equations. Figure 6.59 (left) shows the convergence behavior comparing a ratio of $1 : 5$ compared to $1 : 50$. That is, in the first computation we performed 1 nonlinear multigrid cycle for the mean flow equations compared to 5 for the turbulent flow equations and in the second case we performed 50 multigrid cycles. For the strategy $1 : 50$ the convergence rate is significantly improved. However, compared to the computation with the Spalart-Allmaras model shown in Figure 6.1 the dramatic loss in algorithmical performance cannot be overseen. On the other hand, it is hard to assess if there is an improvement in the accuracy of the results. Though turbulence modeling and the assessment of turbulence models is not topic of this thesis, to the author's opinion the discussion is worthwhile if an increase in the complexity of a turbulence model and the introduced stiffness into the system of equations is justified in case one cannot obtain significant better results.

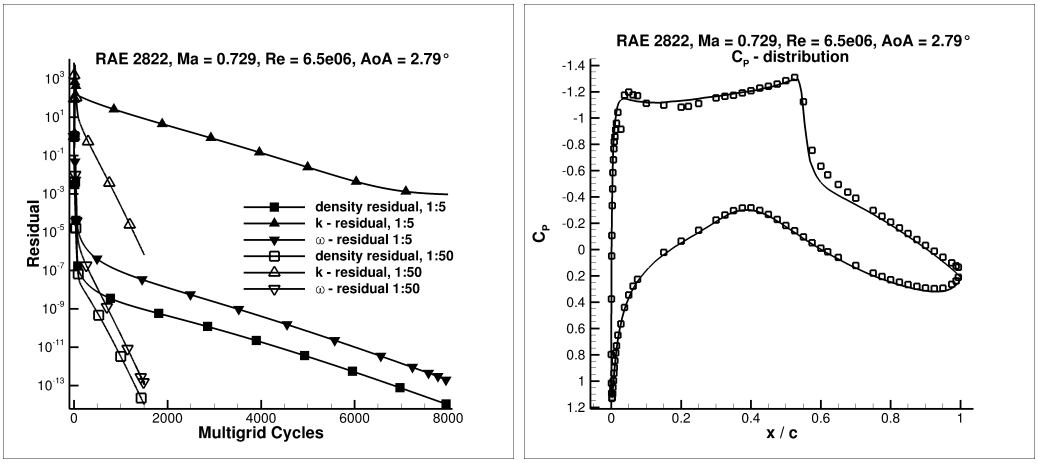

Figure 6.59: Left: Comparison of convergence histories, right: C_P–distribution

This example demonstrates that compared to the solution method for the mean flow equations the multigrid cycle for the turbulent flow equations for the $k\omega$-model is much less efficient. Solving the equations in weakly coupled fashion we may exploit the possibility to increase artificially the computational effort for the turbulent flow equations. Nevertheless, though treating the turbulent flow equations implicitly a

much more effective solution method is required to come to comparable results to the Spalart-Allmaras model. For this turbulence model a residual reduction of 14 orders of magnitude was possible in about 100 multigrid cycles (see Section 6.3), which reduces the computational time significantly. To find much more effective solution methods for two equation $k\omega$ models for unstructured grid solvers is future work. Finally Figure 6.59 (right) shows the determined C_p–distribution, which is in agreement with the measurements. Note, the computation was done fully turbulent.

6.11.3 Example 14: MDA30P30N

To demonstrate the applicability of the $k\omega$-model to a high-lift flow, we reconsider the test case of Section 6.4. Here we only show the computation on the finest mesh described in Table 6.2. Again, the physical parameters of the test case are as follows (see [37]):

- Geometry: MDA30P30N airfoil
- Reynolds number: $Re = 9.0 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.2$
- Angle of attack: AoA = 16.0 °.

Turbulence is modeled using the $k\omega$-model (2.52). Though it is possible to reduce the residual 14 orders of magnitude, which is shown in Figure 6.60 (left), for the $k\omega$-model is takes several more iterations and computational time when compared with the Spalart-Allmaras model to obtain a solution (see Figure 6.4. The convergence history for lift and drag coefficients is given in Figure 6.60 (right).
For the $k\omega$-model the C_p distribution presented in Figure 6.61 (right) shows good agreement with experimental data. A plot of the eddy viscosity is given in Figure 6.61 (right). One can observe that the maximum value is attained as expected behind the airfoil. Finally, for additional assessment of the results velocity profiles are plotted for the Spalart-Allmaras model and the $k\omega$-model and compared with experimental data in Figure 6.62. The results signed with SOLAR are computed on the finest mesh described in Table 6.2. Additionally, to get an idea of possible mesh effects, an additional Spalart-Allmaras model result for a structured mesh is plotted. Again, both models the Spalart-Allmaras and the $k\omega$-model give similar results. Nevertheless, the velocity profiles downstream are not well predicted by both models.

6.11.4 Example 15: Transonic turbulent flow over a Common Research Model

To illustrate the applicability of implicit algorithms in combination with the $k\omega$-model for 3D configurations we show a result for the Common Research model of

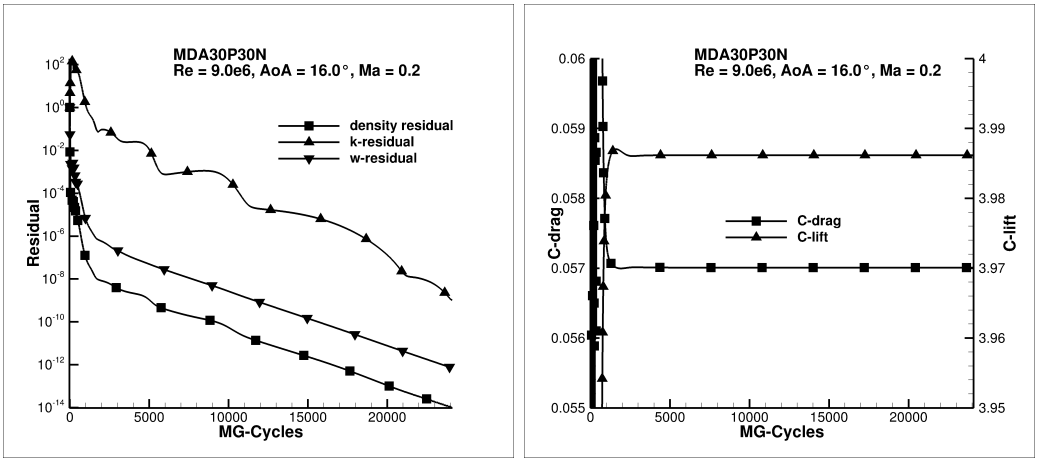

Figure 6.60: Left: Convergence history of residuals, right: Convergence histories of drag and lift coefficients

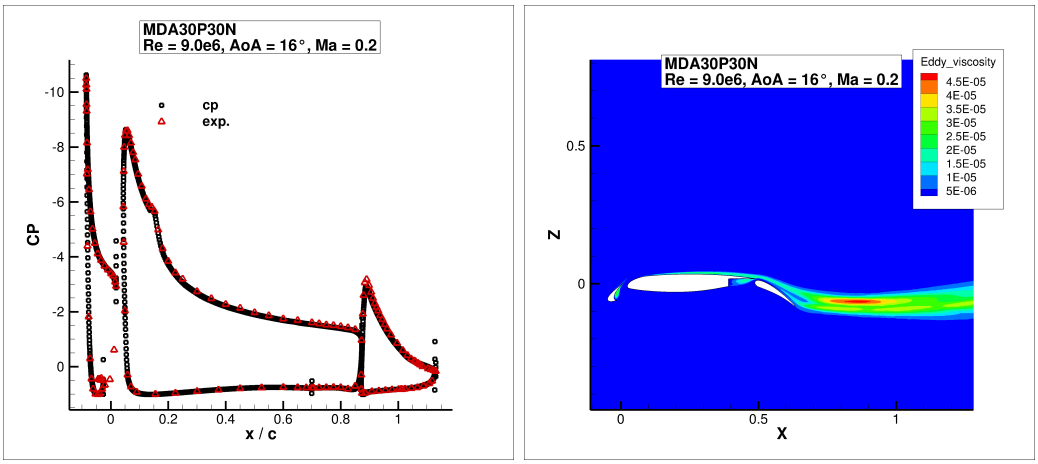

Figure 6.61: Left: C_p distribution for MDA30P30N, Right: Plot of eddy viscosity

the fifth AIAA Drag Prediction workshop already considered in Section 6.4. Due to the complexity to solve for these kinds of equations we restrict ourselves to the hybrid L3 mesh described in Table 6.4. The physical parameters of the test case are as follows:

- Geometry: Wing-body configuration, fifth AIAA Drag Prediction Workshop
- Reynolds number: $Re = 5.0 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.85$
- Angle of attack: AoA $= 2.2\,^\circ$.

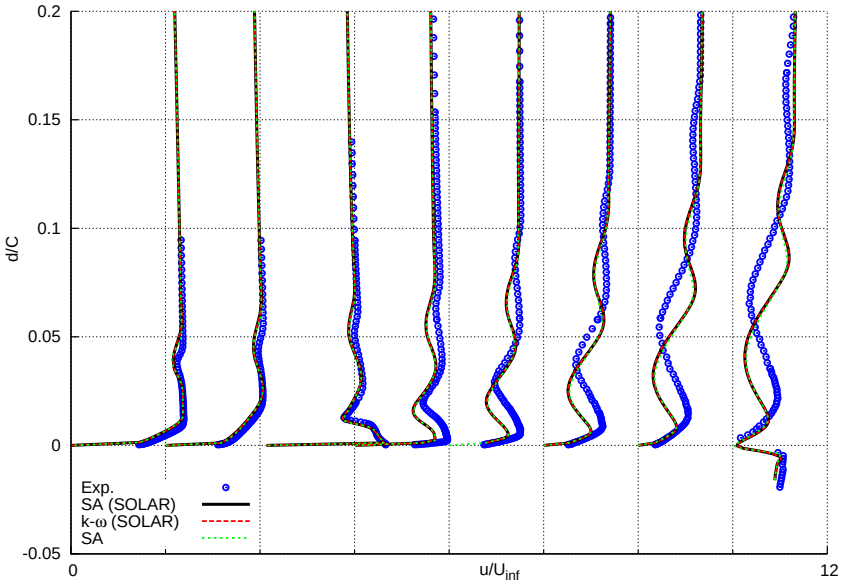

Figure 6.62: Left: Velocity profiles for MDA30P30N

Turbulence is modeled using the $k\omega$-model (2.52). Figure 6.63 (right) visualizes the surface mesh and Figure 6.63 (left) the convergence history. Using consequently implicit solution algorithms a fully converged solution is obtained.

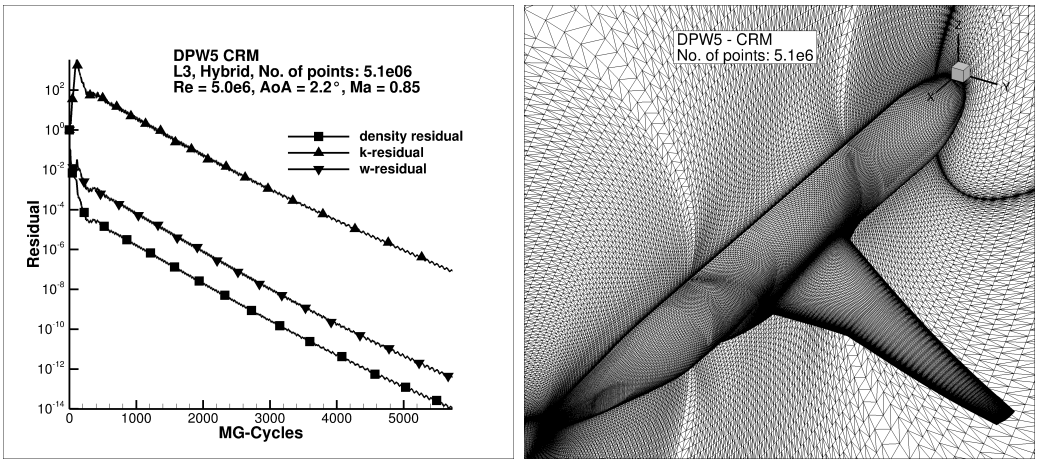

Figure 6.63: Left: Convergence history, Right: Surface mesh of DPW5

6.11.5 Example 16: NASA TRAP Wing

As a final example in this thesis, we show the application of the $k\omega$-model to the NASA TRAP Wing. This test case was already considered in Section 6.9.4. To show the application of the $k\omega$-model we restricted ourselves to the medium mesh described in Table 6.13 and only the 28°case. The physical parameters of the test case are as follows:

- Geometry: NASA TRAP Wing
- Reynolds number: $Re = 4.3 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.2$
- Angle of attack: AoA = 28.0 °.

Turbulence is modeled using the $k\omega$-model (2.52). This example is used to demonstrate that the suggested implementation of the $k\omega$-model together with an implicit solution method has the potential to find fully converged solutions even for 3D high-lift configurations, which exhibit in general large regions of separation and represent a particular challenge for a solution algorithm.

Figure 6.64 (left) shows the convergence history of the residuals, and Figure 6.64 (right) the corresponding convergence of lift and drag coefficients. Below, Figure 6.65 (left) illustrates the C_p distribution on the surface of the wing-body configuration, and Figure 6.65 (right) shows the determined eddy viscosity in the symmetry plane.

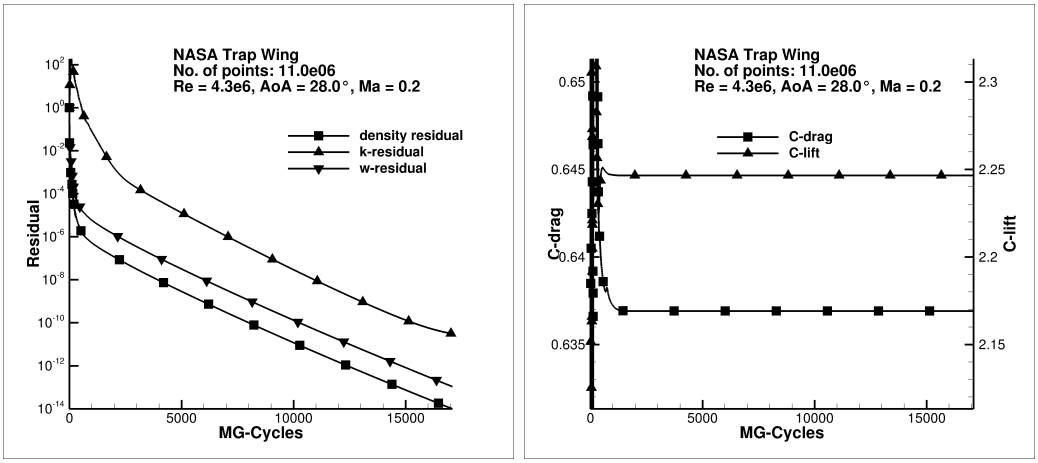

Figure 6.64: Left: Convergence history of residuals, right: Convergence histories of drag and lift coefficients

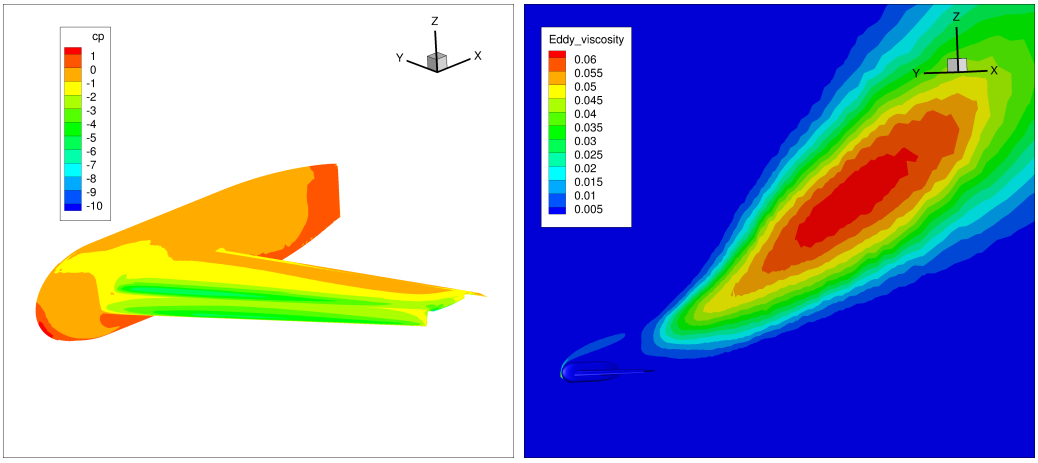

Figure 6.65: Left: C_p distribution on the surface, right: Plot of eddy viscosity

6.12 Summary of numerical examples

It was the goal of this section to demonstrate that the methods developed and suggested in this thesis can be applied to a variety of different flows and fully converged solutions can be obtained. Moreover, it was the intention to demonstrate that the methods are successful even in the case when systematically the number of degrees of freedom are increased.

Nevertheless, it is important to emphasize that an increase of the number of degrees of freedom together with an increase of the complexity of the simulated flow, requires an significant additional effort in the solution methods. On the one hand, this is observed by the fact that in general low cost smoothers suggested in Section 5.3.1 did not have within the author's implementation the potential to find fully converged solutions. Examples for this fact are given in this Section 6. On the other hand, the regularized Newton schemes suggested show a tremendous additional effort per iteration.

This scenario ends in a very difficult situation. Fully converged solutions may only be obtained for regularized Newton kind schemes. But the computational time per iteration might not be acceptable, in particular not when one considers the situation that not only one or a few results are required for demonstration purposes such as in this thesis, but in an industrial environment plenty of such results are needed.

Hence, a significant algorithmical speed up is required. This is due to the fact that the author of this thesis has serious doubts that significant speed up can be expected from the evolution of computer technology in future. In particular the investigation in Section 7.1 in Chapter 7 raise serious doubts that such a significant speed up required can be obtained by modern hardware clusters. The investigations in Section 7.1 give evidence that in the sense of strong scaling, that is the use of

more resources for a fixed problem size, a saturation of speed up is reached rather early. Then the increase in speed up does not legitimate the increase in cost.

With respect to weak scaling, that is an increase in the number of degrees of freedom together with a usage of more resources, the examples considered in this section gave evidence that algorithmical requirements are not yet ready to deal with such an increase in the number of degrees of freedom. Then, the non scaling of the mathematical algorithms is one key issue.

Chapter 7

Assessment of algorithms

The assessment of a solution algorithm for nonlinear equations is a difficult task, in particular if almost none theoretical results are available. As pointed out in the Introduction, for the governing equations of interest in this thesis, the compressible RANS equations and the corresponding boundary value problems formulated in Section 2.4, there is a severe lack of theoretical understanding. Even when we assume that a unique solution exists, we do not have any clue about smoothness of the solution. But in general smoothness is an assumption required to prove convergence of a solution algorithm, and furthermore to prove convergence rates.

Based on the hierarchy of smoothers presented in Section 5.3 we have a certain idea of the potential of the smoothers. Based on the examples shown, in particular in Section 6.7 and Section 6.6.3 it is clear that simplifications of smoothers have severe impact on the obtained solution algorithm based on the nonlinear multigrid Algorithm 5.1.6. Nevertheless, even if we have some heuristic measure for the smoother, it is open in which way it actually acts in the multigrid algorithm. Here, the interaction with the construction of coarse grid and transfer operators plays in important role. And the understanding of these interactions is limited.

Due to the fact that our understanding of the solutions algorithms is very restricted and mainly based on heuristic considerations, it is of major importance to have analysis tools available yielding deeper insight into the behavior of the algorithms. It is the goal of Section 7.2 to develop and apply such a method to some of the examples considered in Chapter 6.

A second topic, which has not been touched in this thesis so far, is the parallel scalability of the suggested algorithms on modern high performance computer clusters. Therefore, note that the key element of the solution algorithms is the application of one of the methods (5.35) – (5.38). The topic of parallel scalability is of major importance as it can be assumed that future cluster hardware is constructed using multicore architectures. Hence, computational speed up with respect to computer hardware can only be expected and obtained in case one can exploit multicore architectures. Section 7.1 is dedicated to this topic.

7.1 Scalability investigations

For several examples in Chapter 6 we have shown the necessity of implicit smoothers for the nonlinear multigrid given by Algorithm 5.1.6. No matter if the smoother is some regularized Newton method described by Algorithm 3 or by some low cost variant (5.41) or (5.42), the key element is the application of the iterations (5.35) – (5.38). In Section 6.3 it was identified that the most expensive operation is the construction of the preconditioner. Freezing the operator on the first stage saved about half of the computational time. A further severe question is the behavior of a solution algorithm, which is based on the the iterations (5.35) – (5.38), with respect to parallelization. Such question has at least two facets:

a) Since we do not use multi-coloring algorithms, application of algorithm (5.36) is line Gauss-Seidel only at the level between partitions. Fixing the number of degrees of freedom and increasing the number of partitions, algorithm (5.36) is getting closer and closer to algorithm (5.35) and even (5.37).

b) Application of one of (5.35) – (5.38) requires after each iteration a communication. Again, fixing the number of degrees of freedom and increasing the number of partitions increases significantly the communication requirements. And moreover, the block sparse matrices become small on each domain. Such imbalance between computational complexity on each domain going hand in hand with an increase in communication may effect the overall efficiency.

The following investigation has the goal to get an idea for the relationship of algorithmical efficiency and parallel scalability.

To this end the test case of Section 6.6 for the L2 hybrid mesh is considered. A rough description of the mesh is given in Table 6.4. The physical parameters of the test case are as follows:

- Geometry: Wing-body configuration, fifth AIAA Drag Prediction Workshop
- Reynolds number: $Re = 5.0 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.85$
- Angle of attack: AoA $= 2.2\,^\circ$.

Turbulence is modeled using the Spalart-Allmaras model (2.52). Hence, the number of degrees of freedom are $6 \cdot 2204089$. However, in the plots we show we neglect the constant factor 6 representing the degrees of freedom per node.

The computational mesh was partitioned using the so-called "Zoltan" library [15]. The number of partitions considered are

$$3 \cdot 2^n, \qquad n = 0, 1, \ldots, 9.$$

The computations we done on the $C^2A^2S^2E$-Cluster of the Institute of Aerodynamics and Flow Technology of the German Aerospace Center. Table 7.1 gives an overview of the number of partitions, the average number of degrees of freedom per partition, the number of computational nodes together with the number of computational cores.

n	No. of partitions	av. NDOF / partition	CPU nodes	CPU kernels
0	3	733333	1	3
1	6	366666	1	6
2	12	183333	1	12
3	24	91666	1	24
4	48	45833	2	48
5	96	22916	4	96
6	192	11458	8	192
7	384	5729	16	384
8	768	2864	32	768
9	1536	1432	64	1536

Table 7.1: Number of partitions and average number of degrees of freedom

The investigated solution algorithm is the one of Theorem 5.3.1 together with (5.42). The algorithm was stopped as soon as (6.3) is satisfied. In detail, the different settings are:

M1
- Algorithm 5.1.6 using a 4v cycle
- Runge-Kutta scheme (6.1)
- Smoother of Theorem 5.3.1 together with 3 symmetric sweeps of (5.36) and $\omega_r = 1$
- Final CFL number: 1000

M2
- Algorithm 5.1.6 using a 4v cycle
- Runge-Kutta scheme (6.1)
- Smoother of Theorem 5.3.1 together with 3 symmetric sweeps of (5.38) and $\omega_r = 1$
- Final CFL number: 1000

M3
- Algorithm 5.1.6 using a 4v cycle
- Runge-Kutta scheme (6.1)
- Smoother of Theorem 5.3.1 together with 6 sweeps of (5.35) and $\omega_r = 1$
- Final CFL number: 1000

M4
- Algorithm 5.1.6 using a single grid iteration
- Runge-Kutta scheme (6.1)
- Smoother of Theorem 5.3.1 together with 6 symmetric sweeps of (5.36) and $\omega_r = 1$
- Final CFL number: 200

M5
- Algorithm 5.1.6 using a single grid iteration
- Runge-Kutta scheme (6.1)
- Smoother of Theorem 5.3.1 together with 6 symmetric sweeps of (5.38) and $\omega_r = 1$
- Final CFL number: 200

Table 7.2 shows on the left the number of iterations required to hit the truncation criterion (6.3). For this investigation this is only of minor interest. The numbers of interest are on the right. These represent the total computational time. Table 7.3 gives additionally the average computational time per iteration. It can be

n	M1	M2	M3	M4	M5	M1	M2	M3	M4	M5
3	192	349	408	3195	4617	48.1	83.5	106.5	185.7	199.3
6	192	344	408	3367	4758	25.2	43.4	55.8	108.2	109.4
12	190	339	415	3319	4691	14.3	24.5	32.5	58.8	61.2
24	193	334	416	3289	4621	8.98	14.9	20.1	35.5	37.9
48	201	334	418	3335	4606	4.99	7.89	10.7	19.4	20.0
96	200	331	425	3237	4575	2.59	4.04	5.72	10.1	9.85
192	211	330	435	3355	4308	1.54	2.20	3.25	6.28	5.49
384	214	327	430	3436	4270	1.02	1.42	2.09	4.48	3.98
768	222	328	439	3575	4272	1.03	1.32	2.05	5.14	4.31
1536	232	—	—	—	—	1.16	—	—	—	—

Table 7.2: Left: No. of iterations, Right: Computational time ($\times$1000 sec.)

directly observed by computational time that while for a moderate number of partitions, that is up to 96 or 192 partitions the speed up due to parallelization might be acceptable. This corresponds to about 20000 degrees of freedom per domain. Using more domains we observe a significant loss in the parallel efficiency. This observation is emphasized using a logarithmic plot indicating the parallel efficiency. Figure 7.1 (left) shows the obtained speed up, and Figure 7.1 (right) the parallel efficiency. These logarithmic plots present that the parallel efficiency already for less than 50000 degrees of freedom per domain is only at about 1/2, for less than about 3000 degrees of freedom per domain it is less than 1/10. Hence, the implementation of the algorithms has severe scalability problems. Denoting the

n	M1	M2	M3	M4	M5
3	250.7	240.0	261.0	58.1	43.2
6	131.1	126.0	136.8	32.2	23.0
12	75.3	72.1	78.4	17.7	13.1
24	46.5	44.7	48.3	10.8	8.20
48	24.8	23.6	25.7	5.83	4.34
96	13.0	12.2	13.5	3.11	2.25
192	7.27	6.67	7.48	1.87	1.28
384	4.76	4.34	4.87	1.30	0.93
768	4.64	4.03	4.67	1.43	1.01
1536	5.01	—	—	—	—

Table 7.3: Average computational time for one iteration in sec.

parallel efficiency by pe, the required resources to obtain the same result behave like $\frac{1}{pe}$. Hence, the potential speed up using further resources has a high price. Such

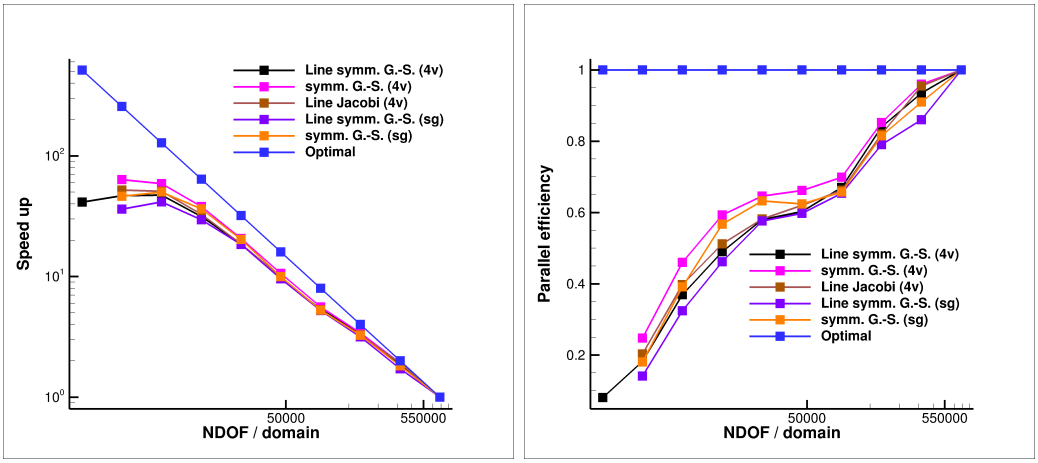

Figure 7.1: Left: Speed up, Right: Parallel efficiency

consideration shows that besides the algorithmical challenges to solve the RANS equations there are additional challenges due to a reasonable exploitation of computational resources. And this effect occurring for strong scalability might not be solved straightforward.

Of course, this is only one initial investigation with respect to strong scalability. But it can be assumed that similar effects are observed for other examples. Future work may focus to identify the major effects contributing to the loss in parallel efficiency. Nevertheless, as long as there are operations inside the algorithm which cannot be (efficiently) parallelized, the author of this thesis assumes that the qualitative

behavior of any such algorithms will look like Figure 7.1. But maybe it is possible to carry the effect over to a smaller number of degrees of freedom.

This investigation shows that investment in computational hardware does not necessarily lead to speed up in computational time. Instead, better suited implementations of the algorithms are required for multicore architectures.

7.2 Computer aided analysis

We now turn to the topic to evaluate at least some of the smoothers suggested in literature about computational fluid dynamics to approximate steady state solutions of the turbulent compressible RANS equations. In Chapter 5 we have demonstrated that all these smoothers for the FAS multigrid method given by Algorithm 5.1.6 are based on multistage Runge-Kutta methods. In Theorems 5.3.1 – 5.3.5 an overview of these methods was given. Demonstrated in Section 6.3 the main effort of these methods is the construction of the block sparse matrix together with the block LU-decomposition required in (5.36) to evaluate $\mathbf{Tri}_{L_i}^{-1}$.

So far, the development of these smoothers is based on several heuristics and a deeper understanding of the power as well as the deficiencies is missing. Often it is not even clear for which range of CFL numbers the methods may work, neither the interplay of several parameters is clear. To design both a robust and efficient solution method a deeper theoretical understanding is necessary. In this section it is the goal to develop an analysis tool, which might give more insight in at least some of the properties of such smoothers.

To understand the properties of Algorithm (5.42) we perform an eigenvalue analysis. With this objective we replace (3.96) by its linearized counterpart. Therefore, we assume that $\mathbf{W}^\dagger$ is a solution of (3.97), that is it satisfies $\mathbf{R}\left(\mathbf{W}^\dagger\right) = 0$. Then, for some small disturbance $\|\mathbf{W}\| < \varepsilon$ of $\mathbf{W}^\dagger$ we approximate

$$\begin{aligned}
\frac{d\mathbf{W}(t)}{dt} &= \frac{d\left(\mathbf{W}^\dagger + \mathbf{W}(t)\right)}{dt} = -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{W}^\dagger + \mathbf{W}\right) \\
&\approx -\mathbf{M}^{-1}\left(\mathbf{R}\left(\mathbf{W}^\dagger\right) + \frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^\dagger\right)\mathbf{W}\right) \\
&= -\mathbf{M}^{-1}\mathbf{A}\mathbf{W}(t), \qquad \mathbf{A} := \frac{d\mathbf{R}}{d\mathbf{W}}\left(\mathbf{W}^\dagger\right).
\end{aligned}$$

Then algorithm (5.42) reads

$$\begin{aligned} \mathbf{W}^{(0)} &:= \mathbf{W}^n \\ \mathbf{W}^{(1)} &= \mathbf{W}^{(0)} - \alpha_{j+1,j}\mathbf{Prec}_1^{-1,\mathrm{app}}\mathbf{A}\mathbf{W}^{(0)} \\ &\vdots \\ \mathbf{W}^{(s)} &= \mathbf{W}^{(0)} - \alpha_{s+1,s}\mathbf{Prec}_1^{-1,\mathrm{app}}\mathbf{A}\mathbf{W}^{(s-1)} \\ \mathbf{W}^{n+1} &= \mathbf{W}^{(s)}, \end{aligned} \tag{7.1}$$

and can be expressed by a polynomial expression,

$$\begin{aligned} \mathbf{W}^{n+1} &= q_s\left(\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\right)\mathbf{W}^n, \\ q_s(z) &= 1 + \sum_{j=1}^{s}(-1)^j z^j \Pi_{i=s-j+1}^{s}\alpha_{i+1,i}. \end{aligned} \tag{7.2}$$

Here the expression $\mathbf{Prec}_1^{-1,\mathrm{app}}\mathbf{A}\mathbf{W}^{(j)}$, $j = 0, \ldots, s-1$, indicates, that the linear system

$$\mathbf{Prec}_1\mathbf{h} = \mathbf{A}\mathbf{W}^{(j)}$$

are solved only approximately by the application of one of the iterative solution methods (5.35) – (5.38) in combination with the stopping criterion (5.39). Then, denoting by λ_i the eigenvalues and by $\mathbf{v}_i$ the corresponding normalized eigenvectors of the linear operator $\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}$ we have

$$\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}, \qquad \Lambda := \mathrm{diag}(\lambda_i), \qquad \mathbf{V} := (\mathbf{v}_1, \mathbf{v}_2, \ldots),$$

and hence the representation

$$\mathbf{W}^{n+1} = \mathbf{V}q_s(\Lambda)\mathbf{V}^{-1}\mathbf{W}^n.$$

To show that algorithm (7.1) converges to the unique limit, it is necessary and sufficient (see [84, Chapter 4]) that the spectral radius of $\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}$ satisfies

$$\rho\left(q_s\left(\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\right)\right) = \rho\left(q_s\left(\Lambda\right)\right) < 1. \tag{7.3}$$

Hence, we are left with the task to find a method determining a suitable approximation to the spectrum of $\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}$.

To compute approximations to the eigenvalues of $\mathbf{P}^{-1,\mathrm{app}}\mathbf{A}$ we exploit the GMRES method (see Algorithm 5.2.1) and its close connection to the Arnoldi process. We apply this method to the linear equation

$$\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\mathbf{h} = \mathbf{R}\left(\mathbf{W} + \mathbf{W}^{\dagger}\right). \tag{7.4}$$

which is given by the inner loop of Algorithm 5.2.1:

Algorithm 7.2.1 *Arnoldi method:*

- *Compute* $\mathbf{r}_0 = \alpha_{j+1,j}\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{R}\left(\mathbf{W}+\mathbf{W}^{\dagger}\right)$
- $\beta := \|\mathbf{r}_0\|_2$, $\mathbf{z}_1 := \frac{1}{\beta}\mathbf{r}_0$
- *for* $j = 1, \ldots, m$
  - $\mathbf{w} := \mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\mathbf{z}_j$
  - *for* $i = 1, \ldots j$
    - $h_{i,j} := \langle \mathbf{w}, \mathbf{z}_i \rangle$
    - $\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{z}_i$
  - $h_{j+1,j} = \|\mathbf{w}\|_2, \mathbf{z}_{j+1} = \frac{1}{h_{j+1,j}}\mathbf{w}$
- $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_m), H_m = (h_{i,j})_{1\le i\le j+1, 1\le j\le m}$

Application of m steps of the Arnoldi process gives the decomposition

$$\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\mathbf{Z}_m = \mathbf{Z}_m\mathbf{H}_m + h_{m+1,m}\mathbf{z}_{m+1}\mathbf{e}_m^*, \qquad \mathbf{e}_m := (0,0,\ldots,0,1)^*, \tag{7.5}$$

where $\mathbf{H}_m := (h_{i,j})_{1\le i,j\le m}$ is an upper Hessenberg matrix,

$$\mathbf{Z}_m^*\mathbf{Z}_m = \mathbf{I}_m \qquad \text{and} \qquad \mathbf{Z}_m^*\mathbf{z}_{m+1} = 0.$$

Multiplying equation (7.5) from the left by $\mathbf{Z}_m^*$ gives

$$\mathbf{Z}_m^*\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\mathbf{Z}_m\mathbf{S}_m = \mathbf{S}_m\mathrm{diag}\left(\mu_j\right)_{j=1,\ldots,m}, \qquad \mathbf{S}_m := (\mathbf{s}_1, \ldots, \mathbf{s}_m), \tag{7.6}$$

where we have assumed that

$$\mathbf{H}_m = \mathbf{S}_m\mathrm{diag}\left(\mu_j\right)_{j=1,\ldots,m}\mathbf{S}_m^{-1}$$

is an eigendecomposition of $\mathbf{H}_m$. Therefore, if the GMRES method with inner Arnoldi iteration truncates after a finite number of steps with the exact solution, the following equation holds:

$$\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\mathbf{Z}_m\mathbf{S}_m = \mathbf{Z}_m\mathbf{S}_m\mathrm{diag}\left(\mu_j\right)_{j=1,\ldots,m}. \tag{7.7}$$

Under this assumption we conclude that $\mathbf{Z}_m\mathbf{S}_m$ are eigenvectors of $\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}$ and μ_j are the corresponding eigenvalues. If the term $h_{m+1,m}\mathbf{z}_{m+1}\mathbf{e}_m^*$ in (7.5) does not vanish, we get for the Ritz pairs $(\mu_j, \mathbf{Z}_m\mathbf{s}_j)$, $j = 1, \ldots, m$, the equations

$$\left\|\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\mathbf{Z}_m\mathbf{s}_j - \mu_j\mathbf{Z}_m\mathbf{s}_j\right\| = h_{m+1,m}\left\|\mathbf{z}_{m+1}\mathbf{e}_m^*\mathbf{s}_j\right\| = h_{m+1,m}\left|\mathbf{e}_m^*\mathbf{s}_j\right|. \tag{7.8}$$

The right hand side of (7.8) gives a computable a posteriori error bound for the approximation quality of the Ritz pairs $(\mu_j, \mathbf{Z}_m \mathbf{s}_j)$ in the Krylov subspace

$$\operatorname{span}\left\{\mathbf{z}_1, \mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\mathbf{z}_1, \ldots, \left(\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\right)^{m-1}\mathbf{z}_1\right\} = \operatorname{span}\left\{\mathbf{z_1}, \ldots, \mathbf{z}_m\right\}$$

when compared with the corresponding exact eigenpairs. In all our numerical experiments we plotted only the eigenvalues satisfying $h_{m+1,m}\left|\mathbf{e}_m^* \mathbf{s}_j\right| < 0.1$. Note that in general no assumption can be made which eigenpairs are approximated. Typically, the outliers in spectrum of the operator are well approximated, while eigenvalues in the bulk of the spectrum are harder to approximate. To evaluate the matrix vector products $\mathbf{Ah}$ inside the GMRES method we use the symmetric finite difference

$$\mathbf{Ah} \approx \frac{1}{2\varepsilon}\left\{\mathbf{R}\left(\mathbf{W} + \varepsilon\mathbf{h}\right) - \mathbf{R}\left(\mathbf{W} - \varepsilon\mathbf{h}\right)\right\}, \qquad \varepsilon = 10^{-6}. \tag{7.9}$$

To transform the spectrum we restrict ourselves to the one stage scheme

$$\mathcal{A} := \begin{pmatrix} 1 \end{pmatrix}, \quad b := \begin{pmatrix} 1 \end{pmatrix},$$

the three stage scheme given by (6.1) and the five stage scheme determined by

$$\mathcal{A} := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.0695 & 1 & 0 & 0 & 0 \\ 0 & 0.1602 & 1 & 0 & 0 \\ 0 & 0 & 0.2898 & 1 & 0 \\ 0 & 0 & 0 & 0.5060 & 1 \end{pmatrix}, \quad b := \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

For a motivation of these coefficients we refer to [93]. The coefficients of these schemes determine the coefficients of the polynomial (7.2). Hence, at least for the algorithm (7.1) formulated for the linearized problem we can only expect convergence if the modified spectrum satisfies (7.3). Moreover, these modified eigenvalues govern the asymptotic convergence. Numerical examples below will demonstrate the good correlation between the approximate spectrum and the convergence behavior for the nonlinear equations.

7.3 Numerical results

To show the applicability of our developed method we investigate the correspondence between the spectrum of $q_s\left(\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\right)$ in (7.2) approximated by the eigenvalues of $\mathbf{H}_m$ and the algorithm (5.42) to solve the nonlinear problem. To this end we perform five steps:

a) Compute a steady state solution. The criterion is that the normalized density residual is reduced 14 orders of magnitude. This corresponds to (6.3).

b) Determine an approximate spectrum of the linearized operator at the steady state solution by computation of the eigenvalues of the upper Hessenberg matrix constructed using Arnoldi's method given by Algorithm 7.2.1.

c) Transform the spectrum by the polynomial describing the Algorithm (7.1), that is evaluate the polynomial corresponding to either the one, the three or the five stage scheme.

d) Determine the largest absolute value of the approximate eigenvalues of the operator $q_s\left(\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\right)$.

e) Starting from the steady state solution, apply the Algorithm (5.42) to observe the correspondence to the approximate spectrum.

Note that multigrid does not play a role. The developed investigation method only comprises single grid iterations. Hence, algorithm (5.42) is interpreted as a (single grid) iterative method in this context. Since the analysis developed does not comprise multigrid, the numerical experiments were also done without multigrid.

7.3.1 Laminar flow around NACA0012 airfoil

Recall the example of Section 6.10. We consider laminar flow around a NACA 00012 airfoil. The relevant physical conditions are

- Geometry: NACA 0012 airfoil
- Reynolds number: $Re = 5000$
- Inflow Mach number: $M_\infty = 0.5$
- Angle of attack: AoA $= 1.0\,^\circ$.

The nomenclature of the different methods compared is taken from Theorems 5.3.1 – 5.3.5. This basic test case demonstrates the wide range of applicability of the developed method as well as its accuracy in predicting the behavior of a solution methodology. Table 7.4 gives an overview of the methods considered as well as the largest absolute value of the approximate eigenvalues of the operator $q_s\left(\mathbf{Prec}^{-1,\mathrm{app}}\mathbf{A}\right)$. In case this value is ≥ 1 we expect divergence, otherwise convergence. Note that all predictions behave as expected. The smaller the CFL number and the larger the number of iterations are, the smaller is the computed largest absolute value. Moreover, all our predictions correspond to the numerical behavior of the nonlinear algorithm (5.42) even in case the values are very close to 1. In particular the behavior of the line implicit method is interesting. For a CFL number of 10 the maximum absolute value is 1.0001793147 and close to 1. And indeed, it takes about 5000 iterations (see Figure 7.3 (right)) until the iteration destabilizes. For a CFL number

of 5 the method is stable, shown by Figure 7.3 (right) and indicated by a maximum absolute value of 0.9979751065.
Besides the prediction accuracy of the developed method this test case shows clearly on a quantitative level that strongly simplified methods are expected to work only with a very limited CFL number. This also holds true in case the iterative linear solution method is too simplified. Both additional features

1) Gauss-Seidel instead of Jacobi and

2) using lines along directions of strong coupling

have a positive effect of the overall behavior. This is observed in both numerical experiments as well as the predicted behavior. Figure 7.4 (left) gives an indication of the effects of lines, the number of sweeps as well as the symmetric Gauss-Seidel sweep to the approximate spectrum. As expected Figure 7.4 (left) shows that the stronger the solution algorithm is formulated, the more clustered are the eigenvalues and an improved convergence behavior can be assumed.

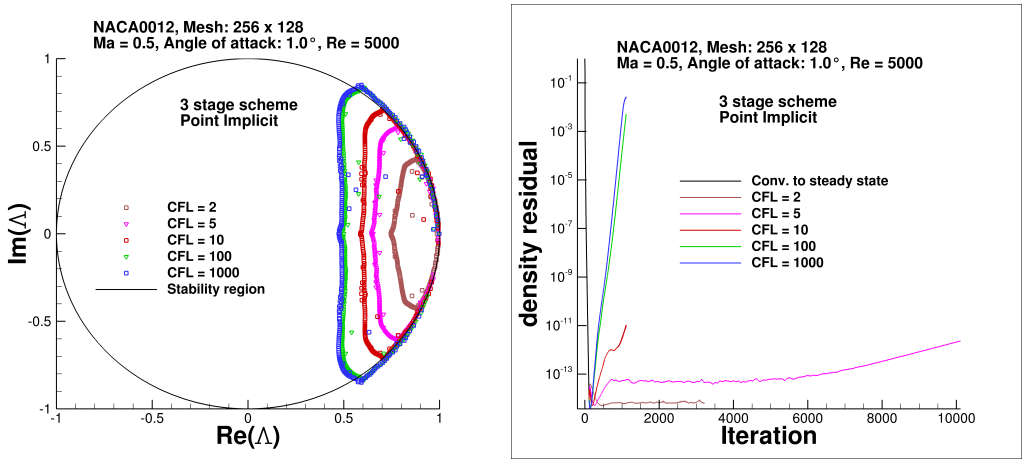

Figure 7.2: Left: Approximate spectrum for the point implicit method and the three stage scheme, Right: Convergence histories for the application of the point implicit method and different CFL numbers.

7.3.2 Turbulent flow around DPW5 CRM

Recall the example of Section 6.6. To show the applicability of the method to large scale 3D problems, we apply the method to the wing-body configuration considered at the fifth AIAA drag prediction workshop. The mesh is the hexahedral L5 mesh with 41231169 number of points (see Table 6.4 and [110]. The relevant physical conditions are

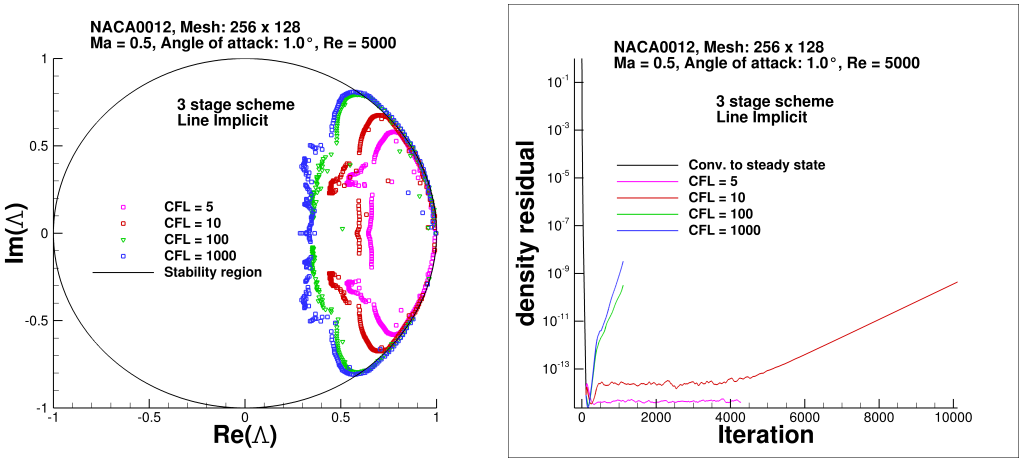

Figure 7.3: Left: Approximate spectrum for the line implicit method and the three stage scheme, right: Convergence histories for the application of the line implicit method and different CFL numbers.

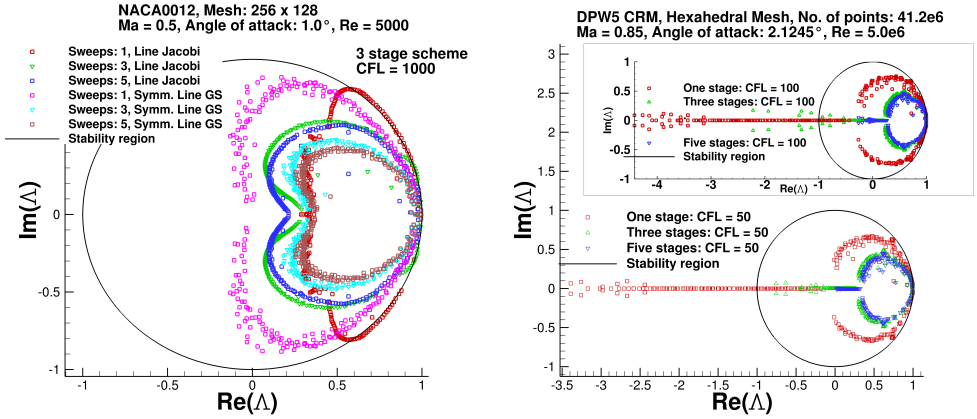

Figure 7.4: Left: Approximate spectrum for the line Jacobi and the symmetric line Gauss-Seidel method with the three stage scheme, right: Approximate spectrum for the DPW5 CRM test case for different multistage schemes.

- Geometry: Wing-body configuration, fifth AIAA Drag Prediction Workshop
- Reynolds number: $Re = 5.0 \cdot 10^6$
- Inflow Mach number: $M_\infty = 0.85$
- Angle of attack: AoA = 2.1245 °. (corresponding to target $C_L = 0.5$, see Table 6.6)

Here we compare different CFL numbers and the one with the three with the five stage scheme. A plot of the different eigenvalue distributions is given in Figure 7.4 (right). Table 7.4 shows that for this test case a restriction of the CFL number is necessary, and that a multistage scheme allows for higher CFL numbers.

Hence, the application of a multistage Runge-Kutta scheme can be interpreted as a stabilization and globalization strategy of Newton's method. This has already been discussed in Section 5.1.7. The analysis tool developed gives now additionally a quantitative assertion. Moreover, this analysis also confirms that a reduction of the CFL number to 50 for the L5 mesh was necessary to obtain a steady state solution (see Section 6.6).

7.4 Considerations for the $k\omega$-model

In Section 5.2.2 it is reported that the derivatives of the destruction terms

$$\frac{\partial De_{k,(k,\omega)}}{\partial k_i} \qquad \text{and} \qquad \frac{\partial De_{\omega,(k,\omega)}}{\partial \omega_i} \tag{7.10}$$

are neglected for the construction of the preconditioner. To give a hint that within our implementation and solution strategy this is necessary, we consider again the example of Section 6.3 for a C-type structured mesh of size 80×16. A fully converged solution was computed satisfying the truncation criterion (6.3).

Let us denote this converged state by $\mathbf{W}^\dagger$. Using this fully converged state $\mathbf{W}^\dagger$ the preconditioner $\mathbf{Prec}_{j,(k,\omega)}$ given by (5.31) is evaluated twice for two different CFL numbers.

a) CFL = 1000, the derivatives of the destruction terms given in (7.10) are included.

b) CFL = 150, the derivatives of the destruction terms given in (7.10) are included.

c) CFL = 1000, the derivatives of the destruction terms given in (7.10) are excluded.

d) CFL = 150, the derivatives of the destruction terms given in (7.10) are excluded.

Then, using Arnoldi's method given by Algorithm 7.2.1 together with the method described in Section 7.2 to determine approximate eigenvalues, an approximate spectrum of $\mathbf{Prec}_{j,(k,\omega)}$ for the four different evaluations described above are computed. The result is plotted in Figure 7.5.
In Figure 7.5 it is shown that the inclusion of the destruction terms leads to eigenvalues which are in the left half plane. In other words, the spectrum has negative

Test case	NACA0012 airfoil		
Method	CFL / No. of sweeps	$\lvert\mu_{max}\rvert$	Numerical experiment
Point. impl.	2/−	0.9977475838	Convergence
	5/−	1.0014868760	Divergence
	10/−	1.0118749482	Divergence
	100/−	1.0319916012	Divergence
	1000/−	1.0349037529	Divergence
Line impl.	5/−	0.9979751065	Convergence
	10/−	1.0001793147	Divergence
	100/−	1.0138678647	Divergence
	1000/−	1.0176708058	Divergence
Jacobi	1000/3	1.0851919018	Divergence
	1000/5	1.0335387144	Divergence
	1000/25	1.0187050259	Divergence
	100/3	1.0651620607	Divergence
	100/5	1.0118067193	Divergence
	100/25	1.0095959280	Divergence
Symm. GS.	1000/1	0.9984234337	Convergence
	1000/3	0.9959707382	Convergence
	1000/5	0.9935652402	Convergence
Line Jacobi	1000/1	1.0176708058	Divergence
	1000/3	1.0035661412	Divergence
	1000/5	0.9897471389	Convergence
Symm. line GS	1000/1	0.9948666248	Convergence
	1000/3	0.9816747477	Convergence
	1000/5	0.9761967171	Convergence
Test case	DPW5 CRM		
One stage scheme and Symm. line GS	1000/5	7.0094725506	Divergence
	100/5	4.4221723489	Divergence
	50/5	3.3905148561	Divergence
Three stage scheme and Symm. line GS	1000/5	12.1781670588	Divergence
	100/5	2.2273552776	Divergence
	50/5	0.9972351764	Convergence
Five stage scheme and Symm. line GS	1000/5	7.0336605534	Divergence
	100/5	0.9934794335	Convergence
	50/5	0.9969464849	Convergence

Table 7.4: Variation of solution methods and their effects on the algorithmic behavior.

parts for both CFL numbers 150 and 1000. As a consequence, none of the considered iterative methods (5.35) – (5.38) was possible to converge. Hence, no suitable update could be obtained.

Neglecting the derivatives of the destruction terms (7.10) in the preconditioner $\mathbf{Prec}_{j,(k,\omega)}$ given by (5.31) the eigenvalues in the left half space vanish. However, the zoom of Figure 7.5 in a neighborhood of the origin shown in Figure 7.6 indicates that for a CFL number of 1000 there are at least two approximate eigenvalues in

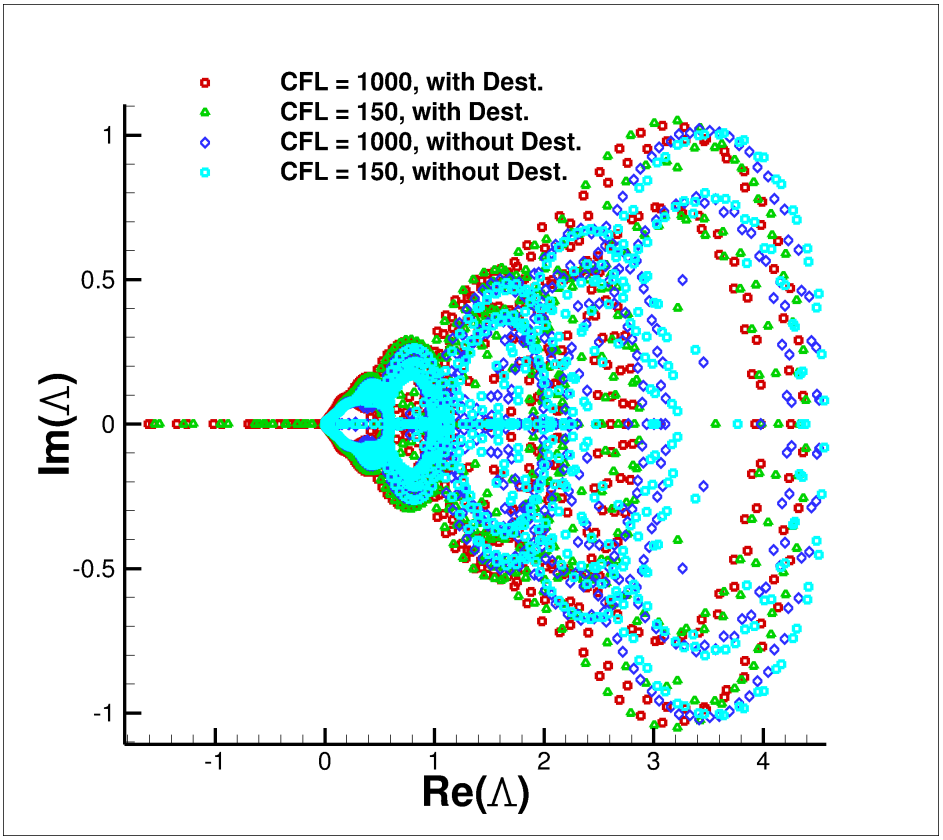

Figure 7.5: Approximate spectrum $\mathbf{Prec}_{j,(k,\omega)}$ for different CFL numbers and inclusion and exclusion of the derivatives of the destruction terms (7.10)

the left half space. Hence, for CFL = 1000 we again observe divergence of the iterative methods (5.35) – (5.38). Reducing the CFL number to 150 the approximate eigenvalues are all located in the right half space and the iterative methods (5.35) – (5.38) converge to an approximate solution yielding an overall stable solution method for this test case.

This investigation has at least two conclusions. Using a full derivative for the $k\omega$-model one possibly obtains matrices which do allow for the application of the iterative solvers (5.35) – (5.38). Hence, further possibly undesired modifications of the derivative are required. Second, even with modifications the maximum achievable CFL number is severely restricted, already for a basic flow case on a coarse mesh.

Naturally, instead of trying to solve the corresponding linear systems by (5.35) – (5.38) we can replace these methods for example by some Krylov subspace method such as GMRES. Such straightforward implementation was considered, but without success yet. As mentioned in Section 5.2.1 a suitable preconditioner is required. Since (5.35) – (5.38) disqualify as preconditioners, the construction of a suitable

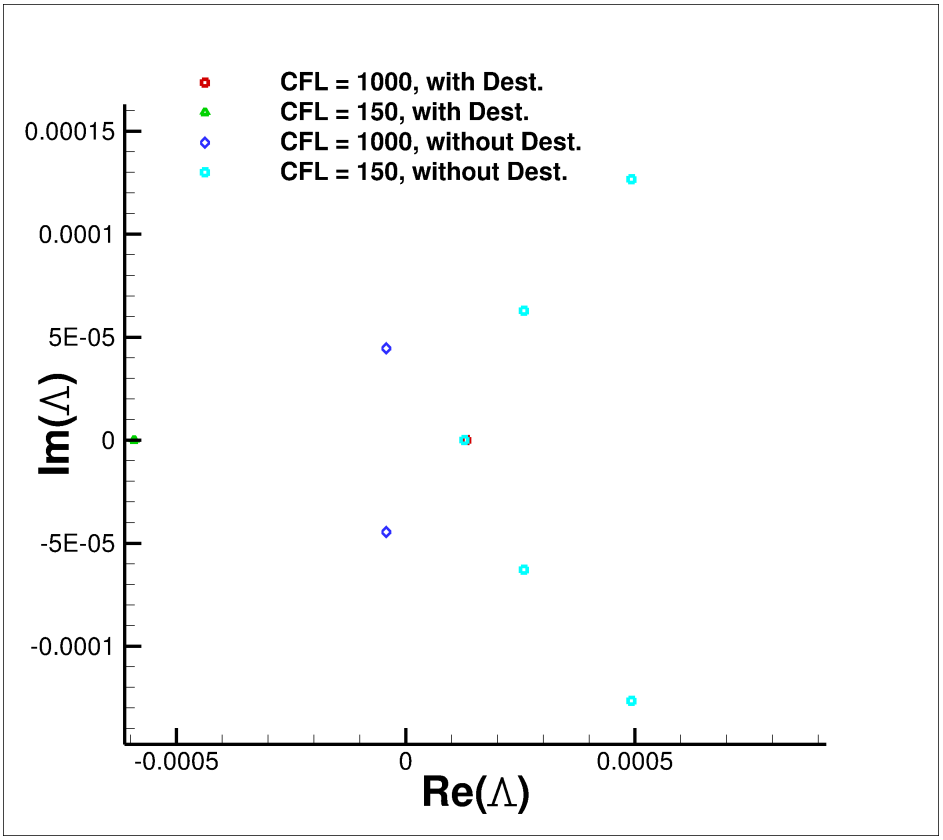

Figure 7.6: Approximate spectrum $\mathbf{Prec}_{j,(k,\omega)}$ for different CFL numbers and inclusion and exclusion of the derivatives of the destruction terms (7.10) in a neighborhood of the origin

preconditioner is open. Considered in Section 6.10, for future work it seems worthwhile to consider in particular for $k\omega$-type models linear multigrid components with suitable smoothers. Maybe such an approach circumvents the problems inherent to these equations.

Chapter 8

Conclusion and Summary

To close this thesis we review the questions mentioned in the Introduction.

a) **Accuracy**: Starting from a full description of the discretization strategy. Are there methods and possibilities to evaluate the accuracy of the approximate solution?

b) **Combination of multigrid with implicit smoother**: What are the main ingredients to construct a reliable algorithm to find approximate solutions of the RANS equations for high Reynolds number turbulent flows?

c) **Application to the incompressible limit**: Are there possibilities to extend the discretization and solution strategy to the incompressible limit to get a closed formulation for the simulation of incompressible and compressible flows, that is to solve the compressible equations for low inflow Mach numbers?

d) **Analysis to assess the methods**: Besides heuristic arguments, are there possibilities to assess the developed methods?

Let us start with a discussion with respect to our findings about **Accuracy**. This investigation goes hand in hand with respect to the **Application to the incompressible limit**. Hence, we unify the answer with respect to these apects. In the framework of an unstructured finite volume code designed to discretize and to find approximate solutions of the compressible Reynolds-averaged Navier-Stokes equations, different strategies are implemented to extend the scheme to the incompressible limit. It has been shown that for several test cases these schemes are successful.

Part of the work was to demonstrate that the low Mach modified schemes work for transonic flow cases with comparable accuracy and robustness to the non-modified schemes. Second, it was the goal of this work to show, especially for high-lift test cases representing flows with large incompressible effects, low inflow Mach number,

and exhibiting strong local compressible effects, that the low Mach modified schemes can be applied successfully.

In particular, the second point fills the gap in the literature that in general low Mach modified schemes for compressible flows are formulated, but they are generally only applied to examples which are globally incompressible. The author's suspicion about this fact, going along with the author's experience, is that the original modified scheme of Turkel works well for globally incompressible flows but shows significant robustness problems for these kinds of mixed problems. Within our implementation, we did not find suitable parameter settings to get fully converged solutions for several test cases including high lift configurations. It should be mentioned that within the family of low Mach modifications (3.51) we only considered one special choice of parameters. Perhaps, other settings improve this situation.

On the other hand, based on the work of Rossow and Swanson, we successfully implemented another low Mach modification to extend a compressible code towards the incompressible limit. This modification worked with comparable robustness to the non-modified scheme. In particular, for several high lift flows we achieved fully converged solutions.

One open point to all the considered schemes is the introduction of the entropy fix. It is a necessary feature for robustness in computing a large number of flows. Neglecting the entropy fix within our implementation, no solution can be obtained in general for a large number of flow cases. In addition, the choice is ad hoc, and it cannot really be justified. Even other choices like a Harten-switch do not circumvent the problem inherent to the introduction of an entropy fix. It can be assumed that further significant improvements of the discretization strategies considered in this thesis can be achieved. Of course, the ultimate objective is to avoid ad hoc fixes and to develop a theoretical foundation to avoid zero artificial viscosity.

In general, one major conclusion of this thesis is, that accuracy of the discretization schemes is hard to assess. To evaluate accuracy we followed in this thesis two different ways. On the one hand, we investigate single data points like pressure distributions or skin-friction distributions and compare these data with expected outcome. On the other hand, from given force coefficients on a sequence of meshes, extrapolation was used to determine possible values for infinitely fine meshes and to compute a corresponding order of convergence. Often, both methods do not agree in the predicted trends. This observation suggests that meshes used were not suited to compute mesh converged solutions.

The accuracy investigations also reveal the following. To judge a solution, it is only of minor interest, and even perhaps misleading, if single data points of a solution are investigated. Future work needs to focus on global convergence criteria. That is, we need to find suitable metrics and norms (e.g. Sobolev norms) to embed the

discrete solution into an adequate function space. The hope is that within such a mathematical setting appropriate convergence criteria can be formulated in a much stricter sense.

From the results obtained, we assume that the following conclusions can be stated. For globally incompressible flow at very low inflow Mach numbers around the NACA 0012 airfoil, the scheme of Rossow/Swanson outperformed all others. For all the other test cases, which are around an inflow Mach number of 0.2 – 0.3, no clear accuracy improvements were found. In particular, for the NASA TRAP wing considered at several angles of attack, no clear assertion can be made, especially for the point at maximum lift, which seems to be somewhere in the region between 32°and 36°. It cannot be excluded that for this test case meshes of significantly higher density (an increase in number of degrees of freedom) are required to get a representative statement. For an even more thorough discussion about accuracy considerations we refer to [46]. The results obtained for the transonic test cases were in the range of expectations. Experimental data such as C_p distributions could be simulated for basic airfoil test cases, wing and wing-body flows. It was demonstrated that TSL gradients may have significant influence on lift and drag coefficients. Nevertheless, though results were obtained for mesh refinement studies, an assessment of the accuracy is hard to obtain.

Second, we report about the necessity of the **Combination of multigrid with implicit smoother**. For an agglomerated FAS multigrid scheme, we have presented an implicit smoothing method derived by a general multistage diagonal implicit Runge-Kutta scheme. It has been shown that this smoother comprises almost all well known smoothing techniques suggested in the literature about computational fluid dynamics. Techniques ranging from explicit local time stepping to regularized Newton methods can be derived using certain parameter choices. We have given evidence for several test cases that regularized Newton methods are in general necessary to obtain fully converged solutions. In particular, with respect to mesh refinement studies, there is strong requirement for implicit smoothers.

We have applied the algorithm successfully to different kinds of grids and grid families. For all considered test cases, machine accuracy could be reached. Moreover, for all test cases an algorithmical scaling between $O(N)$ and $O(N^2)$ was observed when taking CPU times into account. With respect to mesh refinement studies a loss of linear scalability of the algorithm was often observed. It is mentioned that, in the author's opinion, loss of scalability is no surprise and expected. So far there exist no strict proofs that any methodology considered to solve the RANS equations may yield a scalable algorithm. Moreover, when good algorithmical scalability of about $O(N)$ is observed, this can also simply mean that the meshes considered were not fine enough in the sense that not all relevant flow features were resolved.

To improve algorithmical scalability, the deficiencies with respect to a limited number of coarse multigrid levels must be overcome. Here, two ways might be succesful.

On the one hand, the agglomeration strategy discussed in Section 5.1.2 needs further improvement such that a stable discretization on coarse grids is possible. On the other hand, maybe it is possible to improve the transfer operators presented in Section 5.1.4. In any case, to the author's point of view, there exists a straighforward idea for significant improvements.

With respect to computational scalability and parallelization, that is basicly the domain decomposition, it was shown in Section 7.1 that there is strong demand for improvements. Though the algorithmical necessity of implicit smoothers to approximately solve the RANS equations is obvious, the approach for a suitable parallel implementation is open. With respect to future exploitations of possibly massive parallel computer architectures, an approach for significant better computational scalability is required. Since the main effort of the algorithm comes from the application of block sparse matrices, it might be worthwhile to investigate and try to improve these implementations.

To give an answer to the question for an **Analysis to assess the methods** we have followed different strategies. First of all, in Section 5.3 we discussed the connection of several well known smoothing techniques in computational fluid dynamics. Such discussion gives a good overview of simplifications inherent to each of the methods. For an even better understanding, an analysis tool based on an approximate eigenvalue spectrum is suggested in Section 7.2 .

Based on the linearized RANS equations, a method is developed to evaluate several different solution methods proposed in the literature about computational fluid dynamics. The method is based on Arnoldi's algorithm. It is exploited to determine an approximation of the spectrum to the original operator in the corresponding Krylov subspace. Though several assumptions are made, the examples considered in Section 7.3 show a good correlation between the predicted behavior of the solution method and its behavior in the numerical experiment. Instead of considering a simplified model problem like the classical Fourier analysis, the method can be used to directly investigate the problem of interest.

However, so far the method has several deficiencies. First of all it is used here only in the context of an a posteriori method. In particular, it is used to investigate and compare different solution methods. For the problems considered in this thesis and the parameter settings used, it turns out that crude simplifications to the derivative yield a weak preconditioner such that only small CFL numbers can be reached, already for a basic laminar flow problem on a rather coarse mesh. Within the iterative solution procedure, additional features such as symmetric sweeps, including lines and Gauss-Seidel instead of a Jacobi iteration, are gainful techniques to allow for high CFL numbers. The application of a multistage scheme instead of a one-stage scheme seems to be beneficial if one does not want to put too much effort in solving the linearized systems. In particular, this analysis may be used to optimize stage coefficients for certain kind of problems in future work.

Several future applications for this analysis tool may be considered. First of all, it would be interesting to include the effect of multigrid to the analysis. Second, as already considered in the last example, the design of optimized stage coefficients for certain simulations could be considered. An overall goal might be the introduction of the tool in the daily process of engineer's work to give guidance and better understanding of convergence and robustness problems for complex flow problems.

Bibliography

[1] S. R. Allmaras, F. T. Johnson, and P. R. Spalart, *Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model*, in International Conference on Computational Fluid Dynamics 7, Hawaii, no. ICCFD7-1902 in Conference Proceeding Series, 2012.

[2] K. Asano, *On the Incompressible Limit of the Compressible Euler Equation*, Japan Journal of Applied Mathematics, 4 (1987), pp. 455–488.

[3] B. Baldwin and H. Lomax, *Thin layer approximation and algebraic model for separated turbulent flows*, AIAA Journal 78-257, (1978).

[4] F. Bassi, A. Crivellini, S. Rebay, and M. Savini, *Discontinuous Galerkin solution of the Reynolds-averaged NavierStokes and $k-\omega$ turbulence model equations*, Computers &Fluids, 34 (2005), pp. 507 – 540. Residual Distribution Schemes, Discontinuous Galerkin Schemes and Adaptation.

[5] A. Bertozzi and A. Majda, *Vorticity and Incompressible Flows*, Cambridge U. Press, Cambridge, 2002.

[6] J. Blazek, *Computational Fluid Dynamics: Principles and Applications*, Elsevier Science Ltd., 2001.

[7] G. Bluman and S. Kumei, *Symmetries and Differential Equations*, Applied Mathematical Sciences, Springer New York, 1996.

[8] J. Bredberg, *On Two-equation Eddy-Viscosity Models*, Internal report, Department of Thermo and Fluid Dynamics, Chalmers University of Technology, Göteborg, Sweden, 2001.

[9] J. S. Cagnone, K. Sermeus, S. K. Nadarajah, and E. Laurendeau, *Implicit multigrid schemes for challenging aerodynamic simulations*, Computer & Fluids, 44 (2011), pp. 314–327.

[10] Y. H. Choi and C. L. Merkle, *The application of preconditioning to viscous flows*, Journal of Computational Physics, 105 (1993), pp. 207–223.

[11] A. J. Chorin, *A Numerical Method for Solving Incompressible Viscous Flow Problems*, Journal of Computational Physics, 2 (1967), pp. 12–26.

[12] P. Constantin, *Some Open Problems and Research Directions in the Mathematical Study of Fluid Dynamics*, in in Mathematics Unlimited-2001 and Beyond, Springer Verlag, 2000, pp. 353–361.

[13] P. H. Cook, M. A. McDonald, and M. C. P. Firmin, *Aerofoil RAE 2822 pressure distributions and boundary layer and wake measurements*, AGARD-AR, 138 (1979).

[14] D. L. Darmofal and P. J. Schmid, *The Importance of Eigenvectors for Local Preconditioners of the Euler Equations*, Journal of Computational Physics, 127 (1996), pp. 346 – 362.

[15] K. Devine, E. Boman, L. Riesen, U. Catalyurek, and C. Chevalier, *Getting Started with Zoltan: A Short Tutorial*, in Proc. of 2009 Dagstuhl Seminar on Combinatorial Scientific Computing, 2009. Also available as Sandia National Labs Tech Report SAND2009-0578C.

[16] P. Eliasson, P. Weinerfelt, and J. Nordström, *Application of a line-implicit scheme on stretched unstructured grids*, in Proc. 47th AIAA Aerospace Sciences Meeting, no. 2009-0163 in Conference Proceeding Series, AIAA, 2009.

[17] J. K. Fassbender, *Improved Robustness for Numerical Simulation of Turbulent Flows Around Civil Transport Aircraft at Flight Reynolds Numbers*, PhD thesis, Technical University of Braunschweig, 2004.

[18] C. L. Fefferman, *Existence and Smoothness of the Navier-Stokes Equation*, Clay Mathematics Institute, (2006), pp. 51–61.

[19] K. J. Fidkowsky, *A High-Order Discontinuous Galerkin Multigrid Solver for Aerodynamic Applications*, Master's thesis, Massachusetts Institute of Technology, 2004.

[20] T. Gerhold, M. Galle, O. Friedrich, and J. Evans, *Calculation of Complex Three-Dimensional Configurations Employing the DLR-Tau-Code*, AIAA Paper 1997–167, 1997.

[21] S. K. Godunov, *A Finite Difference Method for the Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics*, Matematicheskii Sbornik, 47 (1959), pp. 357–393.

[22] H. Guillard and C. Viozat, *On the behaviour of upwind schemes in the low Mach number limit*, Computers & Fluids, 28 (1999), pp. 63 – 86.

[23] J. Hadamard, *Lectures on Cauchy's problem in linear partial differential equations*, Dover publications, New York, 1952.

[24] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer, New York, 1992.

[25] D. Hänel and R. Schwane, *An Implicit Flux-Vector Splitting Scheme for the Computation of Viscous Hypersonic Flow*, in Proc. 27th AIAA Aerospace Sciences Meeting, no. 89-0274 in Conference Proceeding Series, AIAA, January 1987.

[26] D. Hänel, R. Schwane, and G. Seider, *On the Accuracy of Upwind Schemes for the Solution of the Navier-Stokes Equations*, in Proc. 25th AIAA Aerospace Sciences Meeting, no. 87-1105 in Conference Proceeding Series, AIAA, January 1987.

[27] A. Harten and J. M. Hyman, *Self Adjusting Grid Methods for One-Dimensional Hyperbolic Conservation Laws*, Journal of Computational Physics, 50 (1983), pp. 235–269.

[28] R. Hartmann, J. Held, and T. Leicht, *Adjoint-based error estimation and adaptive mesh refinement for the RANS and k-ω turbulence model equations*, J. Comput. Phys., 230 (2011), pp. 4268–4284.

[29] http://www.centaursoft.com.

[30] A. Jameson, *Computational Transonics*, Communications on Pure and Applied Mathematics, 41 (1988), pp. 507–549.

[31] A. Jameson, W. Schmidt, and E. Turkel, *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes*, AIAA Paper, AIAA-81-1259, (1981).

[32] W. P. Jones and B. E. Launder, *The prediction of laminarization with a two-equation model of turbulence*, International Journal of Heat and Mass Transfer, 15 (1972), pp. 301–314.

[33] D. S. Kamenetskiy, J. E. Bussoletti, C. L. Hilmes, V. Venkatakrishnan, L. B. Wigton, and F. T. Johnson, *Numerical Evidence of Multiple Solutions for the Reynolds-Averaged Navier-Stokes Equations*, AIAA Journal, 52 (2014).

[34] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice Hall Professional Technical Reference, 2nd ed., 1988.

[35] S. Klainerman and A. Majda, *Singular Limits of Quasilinear Hyperbolic Systems with Large Parameters and the Incompressible Limit of Compressible Fluids*, Comm. Pure Appl. Math., 34 (1981), p. 481524.

[36] ——, *Compressible and Incompressible Fluids*, Comm. Pure Appl. Math., 35 (1982), p. 629651.

[37] S. M. Klausmeyer and J. C. Lin, *Comparative Results From a CFD Challenge Over a 2D Three-Element High-Lift Airfoil*, NASA Technical Memorandum 112858, 1997.

[38] A. N. Kolmogorov, *The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers*, Dokl. Akad. Nauk., SSSR, 30 (1941), pp. 299–303.

[39] ——, *Equations of turbulent motion in incompressible viscous fluids for very large Reynolds numbers*, Izvestia Academy of Sciences, USSR, 6 (1942), pp. 56–58.

[40] S. Langer, *Investigation of Preconditioning Techniques for the Iteratively Regularized Gauss-Newton Method for Exponentially Ill-Posed Problems*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2543–2559.

[41] S. Langer, *Hierarchy of Preconditioning Techniques for the solution of the Navier-Stokes equations discretized by 2nd order unstructured finite volume methods*, in ECCOMAS 2012 CONFERENCE, Conference Proceeding Series, Sept. 2012.

[42] S. Langer, *Investigation and application of point implicit Runge-Kutta methods to inviscid flow problems*, International Journal for Numerical Methods in Fluids, 69 (2012), pp. 332–352.

[43] ——, *Application of a line implicit method to fully coupled system of equations for turbulent flow problems*, Int. J. Comput. Fluid Dyn., 27 (2013), pp. 131–150.

[44] S. Langer, *Agglomeration multigrid methods with implicit Runge-Kutta smoothers applied to aerodynamic simulations on unstructured grids*, Journal of Computational Physics, 277 (2014), pp. 72 – 100.

[45] ——, *Computer aided analysis of preconditioned multistage Runge-Kutta methods applied to solve the compressible Reynolds averaged Navier-Stokes equations*, in New Results in Numerical and Experimental Fluid Mechanics X: Contributions to the 19th STAB/DGLR Symposium Munich, Germany, 2014, H. P. Kreplin, ed., Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer, 2015.

[46] ——, *Investigations of a compressible second order finite volume code towards the incompressible limit*, Computers & Fluids, 149 (2017), pp. 119 – 137.

[47] S. Langer and D. Li, *Application of point implicit Runge-Kutta methods to inviscid and laminar flow problems using AUSM and AUSM+ upwinding*, International Journal of Computational Fluid Dynamics, 25 (2011), pp. 255–269.

[48] S. Langer, A. Schwöppe, and N. Kroll, *Investigation and Comparison of Implicit Smoothers Applied in Agglomeration Multigrid*, AIAA Journal, 53 (2015), pp. 2080 – 2096.

[49] J. V. Lassaline and D. W. Zingg, *Development of an Agglomeration Multigrid Algorithm with Directional Coarsening*, in AIAA Computational Fluid Dynamics Conference, 14 th, no. 99-3338 in Conference Proceeding Series, 1999.

[50] D. W. Levy, K. R. Laflin, E. N. Tinoco, J. C. Vassberg, M. Mani, B. Rider, C. L. Rumsey, R. A. Wahls, J. H. Morrison, O. P. Brodersen, S. Crippa, D. J. Mavriplis, and M. Murayama, *Summary of Data from the Fifth AIAA Drag Prediction Workshop*, in Proc. 51st AIAA Aerospace Sciences Meeting, Gravepine (Dallas/Ft. Worth Region), Texas, January 2013, no. 2013-0046 in Conference Proceeding Series, AIAA, 2013.

[51] M. S. Liou, *A Sequel to AUSM: AUSM+*, Journal of Computational Physics, 129 (1995), pp. 364–382.

[52] M. S. Liou and J. Steffen, *A New Flux Splitting Scheme*, Journal of Computational Physics, 107 (1993), pp. 23–29.

[53] D. G. Martineau, S. Stokes, S. J. Munday, and A. P. Jackson, *Anisotrpoic Hybrid Mesh Generation for Industrial RANS Applications*, in AIAA Aerospace Sciences Meeting and Exhibit, 44 th, Reno, Nevada, no. 2006-534 in Conference Proceeding Series, 2006.

[54] D. J. Mavriplis, *Multigrid Technqiues For Unstructured Meshes*, ICASE Report 95-27, 1995.

[55] ——, *Directional Coarsening and Smoothing for anisotropic Navier-Stokes Problems*, Electronic Transactions on Numerical Analysis, 6 (1997), pp. 182–197.

[56] ——, *Directional Agglomeration Multigrid Techniques for High-Reynolds Number Viscous Flows*, ICASE Report No.98-7, (1998).

[57] ——, *Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes*, ICASE Report No.98-6, (1998).

[58] A. Meister, *Asymptotic Expansions and Numerical Methods in Computational Fluid Dynamics*, no. 17 in Hamburger Beiträge zur Angewandten Mathematik, Computational Fluid Dynamics and Data Analysis, December 2001.

[59] F. R. Menter, *Improved Two-Equation k–ω Turbulence Models for Aerodynamic Flows*, tech. rep., NASA, 1992.

[60] ——, *Zonal Two Equation $k-\omega$ Turbulence Models for Aerodynamic Flows*, tech. rep., NASA, 1993.

[61] ——, *Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications*, Aiaa Journal, 32 (1994), pp. 1598–1605.

[62] ——, *Review of the shear-stress transport turbulence model experience from an industrial perspective*, International Journal of Computational Fluid Dynamics, 23 (2009), pp. 305–316.

[63] P. Moinier, *Algorithm Developments for an Unstructured Viscous Flow Solver*, PhD thesis, University of Oxford, 1999.

[64] I. Moulitsas and G. Karypis, *Multilevel algorithms for generating coarse grids for multigrid methods.*, in SC, G. Johnson, ed., ACM, 2001, p. 45.

[65] E. J. Nielsen, W. K. Anderson, R. W. Walters, and D. E. Keyes, *Application of Newton-Krylov Methodology to a Three-Dimensional Unstructured Euler Code*, AIAA Paper 95-1733-CP, 1995.

[66] E. J. Nielsen, J. Lu, M. A. Park, and D. L. Darmofal, *An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids*, Computer & Fluids, 33 (2004), pp. 1131–1155.

[67] H. Nishikawa, B. Diskin, and J. L. Thomas, *Critical study of agglomerated multigrid methods for diffusion*, AIAA Journal, 48 (2010), pp. 839–847.

[68] H. Nishikawa, B. Diskin, J. L. Thomas, and D. H. Hammond, *Recent advances in agglomerated multigrid*, in 51st AIAA Aerospace Sciences Meeting, AIAA Paper 2013-863, Texas, 2013.

[69] ——, *Recent advances in agglomerated multigrid*, in 51st AIAA Aerospace Sciences Meeting, AIAA Paper 2013-863, Texas, 2013.

[70] S. Osher and F. Solomon, *Upwind Schemes for Hyperbolic Systems of Conversation Laws*, Mathematics of Computation, 38 (1982), pp. 339–374.

[71] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1980.

[72] O. Peles, E. Turkel, and S. Yaniv, *Fast Iiterative Methods for Navier Stokes Equations with SST Turbulence Model and Chemistry*, in International Conference on Computational Fluid Dynamics 7, Hawaii, no. ICCFD7-1902 in Conference Proceeding Series, 2012.

[73] N. A. Pierce and M. B. Giles, *Preconditioned multigrid methods for compressible flow calculations on stretched meshes*, J. Comput. Phys, 136 (1997), pp. 425–445.

[74] N. A. Pierce, M. B. Giles, A. Jameson, and L. Martinelli, *Accelerating Three-Dimensional Navier-Stokes Calculations*, in AIAA Computational Fluid Dynamics Conference, 13 th, no. 1997-1953 in Conference Proceeding Series, AIAA, 1997.

[75] J. J. Quirk, *A Contribution to the Great Riemann Solver Debate*, NASA Contractor Report 191490, ICASE Report 92-64, (1992).

[76] W. Rodi and D. B. Spalding, *A two-parameter model of turbulence, and its application to free jets*, Wärme und Stoffübertragung, 3 (1970), pp. 85–95.

[77] P. L. Roe, *Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes*, Journal of Computational Physics, 43 (1981), pp. 357–372.

[78] C.-C. Rossow, *Extension of a Compressible Code Toward the Incompressible Limit*, AIAA Journal, 41 (2003), pp. 2379–2386.

[79] ——, *Convergence acceleration for solving the compressible Navier-Stokes equations*, AIAA Journal, 44 (2006), pp. 345–352.

[80] ——, *Efficient computation of compressible and incompressible flows*, Journal of Computational Physics, 220 (2007), pp. 879–899.

[81] C. L. Rumsey, *Apparent transition behavior of widely-used turbulence models*, International Journal of Heat and Fluid Flow, 28 (2007), pp. 1460–1471.

[82] C. L. Rumsey and J. P. Slotnick, *Overview and Summary of the Second AIAA High Lift Prediction Workshop*, AIAA Paper, AIAA-2014-0747, (2014).

[83] C. L. Rumsey, J. P. Slotnick, M. Long, A. R. Stuever, and T. R. Wayman, *Summary of the First AIAA CFD High-Lift Prediction Workshop*, Journal of Aircraft, 48 (2011), pp. 2068–2079.

[84] Y. Saad, *Iterative Methods for Sparse Linear Systems*, International Thomson Publishing, Boston, 1996.

[85] P. R. Spalart and S. R. Allmaras, *A One-Equation Turbulence Model for Aerodynamic Flows*, in AIAA Computational Fluid Dynamics Conference, no. 1992-439 in Conference Proceeding Series, AIAA, 1992.

[86] J. L. Steger and R. F. Warming, *Flux Vector Splitting of the inviscid Gasdynamic Equations with Application to Finite Difference Methods*, Journal of Computational Physics, 40 (1981), pp. 263–293.

[87] B. Stroustrup, *The C++ Programming Language*, Addison-Wesley Professional, 4th ed., 2013.

[88] M. Svärd, J. Gong, and J. Nordström, *Stable artificial dissipation operators for finite volume schemes on unstructured grids*, Applied Numerical Mathematics, 56 (2006), pp. 1481–1490.

[89] M. Svärd and J. Nordström, *Stability of finite volume approximations for the laplacian operator om quadrilateral and triangular grids*, Applied Numerical Mathematics, 51 (2004), pp. 101–125.

[90] R. Swanson and S. Langer, *Steady-state laminar flow solutions for NACA0012 airfoil*, Computers & Fluids, 126 (2016), pp. 102 – 128.

[91] R. C. Swanson and C.-C. Rossow, *An efficient solver for the RANS equation and a one-equation turbulence model*, Computers & Fluids, 42 (2011), pp. 13–25.

[92] R. C. Swanson and E. Turkel, *On central difference and upwind schemes*, Journal of Computational Physics, 101 (1997), pp. 292–306.

[93] R. C. Swanson, E. Turkel, and C.-C. Rossow, *Convergence acceleration of Runge-Kutta schemes for solving the Navier-Stokes equations*, J. Comput. Phys., 224 (2007), pp. 365–388.

[94] R. C. Swanson, E. Turkel, and S. Yaniv, *Analysis of a R/Implicit Smoother for Multigrid*, in Computational Fluid Dynamics 2010: Proceedings of the Sixth International Conference on Computational Fluid Dynamics, ICCFD6, St Petersburg, Russia, on July 12-16, 2010, Springer-Verlag Berlin Heidelberg, 2011, pp. 409–417.

[95] J. L. Thomas, B. Diskin, and H. Nishikawa, *A critical study of agglomerated multigrid methods for diffusion on highly-stretched grids*, Computers & Fluids, 41 (2011), pp. 82 – 93. Implicit Solutions of NavierStokes Equations. Special Issue Dedicated to Drs. W.R. Briley and H. McDonald.

[96] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, Inc., Orlando, FL, USA, 2001.

[97] E. Turkel, *Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations*, Journal of Computational Physics, 72 (1987), pp. 277–298.

[98] E. Turkel, *Improving the accuracy of central difference schemes*, in 11th International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, D. L. Dwoyer and M. Y. Hussaini, eds., vol. 323, Springer-Verlag, 1989.

[99] E. Turkel, *Preconditioning-Squared Methods for multidimensional Aerodynamics*, in 28th AIAA Fluid Dynamics Conference, Snowmass Village, CO, U.S.A., no. 97-2025 in Conference Proceeding Series, June 1997.

[100] E. Turkel, *Preconditioning techniques in computational fluid dynamics*, Annu. Rev. Fuid Mech., 31 (1999), pp. 385–416.

[101] E. Turkel, *Robust Low Speed Preconditioning for Viscous High Lift flows*, in AIAA Computational Fluid Dynamics Conference, no. 2002-0962 in 40th AIAA Aerospace Sciences Meeting & Exhibit Reno, NV, U.S.A., AIAA, 2002.

[102] E. Turkel, A. Fiterman, and B. van Leer, *Preconditioning and the Limit of the Compressible to the Incompressible Flow Equations for Finite Difference Schemes*, in Frontiers of Computational Fluid Dynamics, D. A. Caughey and M. M. Hafez, eds., John Wiley and Sons, 1994, pp. 215–234.

[103] E. Turkel, R. Radespiel, and N. Kroll, *Assessment of Preconditioning Methods for Multidimensional Aerodynamics*, Computers & Fluids, 26 (1997), pp. 613 – 634.

[104] E. Turkel and V. N. Vatsa, *Effect of Artificial Viscosity on Three-Dimensional Flow Solutions*, AIAA Journal, 32 (1994), pp. 39–45.

[105] D. L. Tweedt, R. V. Chima, and E. Turkel, *Preconditioning for Numerical Simulation of Low mach Number Three-Dimensional Viscous Turbomachinery Flows*, in AIAA Computational Fluid Dynamics Conference, no. 1997-1828 in 28th Fluid Dynamics Conference Snowmass Village, CO, U.S.A., American Institute of Aeronautics and Astronautics, 1997.

[106] S. Ukai, *The Incompressible Limit and the Initial Layer of the Compressible Euler Equation*, Journal of Mathematics of Kyoto University, 26 (1986), pp. 323–331.

[107] B. Van Leer, *Flux-Vector Splitting for the Euler Equations*, ICASE Report 82-30, (1982).

[108] B. van Leer, *Flux-vector splitting for the euler equations*, in Eighth International Conference on Numerical Methods in Fluid Dynamics: Proceedings of the Conference, Rheinisch-Westfälische Technische Hochschule Aachen, Germany, June 28 – July 2, 1982, E. Krause, ed., Springer Berlin Heidelberg, Berlin, Heidelberg, 1982, pp. 507–512.

[109] B. van Leer, W.-T. Lee, and P. L. Roe, *Characteristic time-stepping or local preconditioning of the Euler equations*, in 10th Computational Fluid Dynamics Conference, 1991, pp. 260–282.

[110] J. C. Vassberg, *A Unified Baseline Grid about the Common Research Model Wing-Body for the Fifth AIAA CFD Drag Prediction Workshop*, in Proc. 29th AIAA Applied Aerodynamics Conference, Honolulu, HI, June 2011, no. 2011-3508 in Conference Proceeding Series, AIAA, 2011.

[111] C. Viozat, *Implicit Upwind Schemse for Low Mach Number Compressible Flows*, Institut National de Recherche en Informatique et en Automatique, INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France), ISSN 0249-6399, 1997.

[112] D. C. Wilcox, *Reassessment of the Scale-Determining Equation for Advanced Turbulence Models*, AIAA Journal, 26 (1988), pp. 1299–1310.

[113] ——, *Turbulence Modeling for CFD*, DCW Industries, Incorporated, 1994.

[114] ——, *Formulation of the k-w turbulence model revisited*, AIAA Journal, 46 (2008), pp. 2823–2838.

[115] J. S. Wong, D. Darmofal, and J. Peraire, *The Solution of the Compressible Euler Equations at low Mach Numbers using a stabilized Finite Element Algorithm*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 5719–5737.

[116] S. Yoon and A. Jameson, *Lower-upper Symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations*, in 25th AIAA Aerospace Sciences Meeting, AIAA Paper 1987-0600, Reno, NV, 1987.